Identification of Transition Models of Biological Systems in the Presence of Transition Noise

Ashwin Srinivasan[1], Michael Bain[2], Deepika Vatsa[3], and Sumeet Agarwal[3]

[1] Birla Institute of Technology and Science Pilani, Goa Campus
[2] School of Computer Science and Engineering, University of New South Wales, Sydney
[3] Department of Electrical Engineering, Indian Institute of Technology Delhi
ashwin@goa.bits-pilani.ac.in,
m.bain@unsw.edu.au,
{eez138262,sumeet}@iitd.ac.in

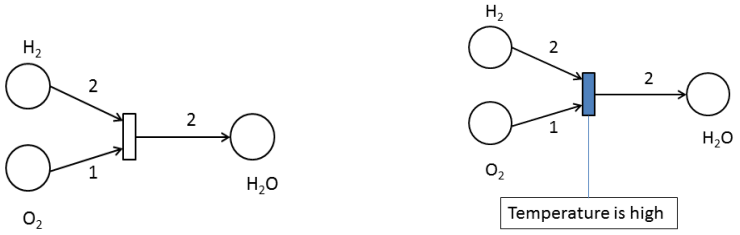
Abstract. The identification of transition models of biological systems (Petri net models, for example) in noisy environments has not been examined to any significant extent, although they have been used to model the ideal behaviour of metabolic, signalling and genetic networks. Progress has been made in identifying such models from sequences of qualitative states of the system; and, more recently, with additional logical constraints as background knowledge. Both forms of model identification assume the data are correct, which is often unrealistic since biological systems are inherently stochastic. In this paper, we model the transition noise that can affect model identification as a Markov process where the corresponding transition functions are assumed to be known. We investigate, in the presence of this transition noise, the identification of transitions in a target model. The experiments are re-constructions of known networks from simulated data with varying amounts of transition-noise added. In each case, the target model traces a specific trajectory through the state-space. Model structures that explain the noisy state-sequences are obtained based on recent work which formulates the identification of transition models as logical consequence-finding. With noisy data, we need to extend this formulation by allowing the abduction of new transitions. The resulting structures may be both incorrect and incomplete with respect to the target model. We quantify the ability to identify the transitions in the target model, using probability estimates computed from transition-sequences using PRISM. Empirical results suggest that we are able to identify correctly the transitions in the target model with transition noise levels ranging from low to high values.

1 Introduction

Chemical equations are symbolic statements not of what will happen, but of what may happen. Thus, the equation $2H_2(g) + O_2(g) \rightarrow 2H_2O(g)$ does not mean that hydrogen and oxygen will necessarily react to produce water (the "g" denotes the reactants and products are in a gaseous state). Filling a balloon with

the reactants, for example, does not immediately result in balloon full of water vapour. Additional conditions may be needed (in this case, a high temperature) for the reaction to occur. Even if external conditions are favourable, it is possible that the reaction may not proceed. There is thus a non-determinism associated with any chemical processes, including those that occur in cells [6].

Our interest in this paper is in computational models of biological networks, like Petri nets, in which processes are *transitions* representing local changes to the qualitative state of the system. An example of a Petri net representation of the "water" reaction is in Fig. 1a.



(a) A simple Petri net. (b) A simple LGTS.

Fig. 1: Two transition system representations of the reaction $2H_2 + O_2 \rightarrow 2H_2O$.

We extend our previous work to study identification of transition models of biological systems in the presence of added transition noise. We develop a two-stage method (Fig. 2). First, deductive and abductive inference is used in a logic programming framework [17] to identify all transitions consistent with observational data. Second, probabilistic logic programming [15] is used to estimate parameters for the identified transition system. Evaluation is by reconstruction experiments on benchmark biological systems with varying levels of added transition noise. In Section 2 we describe our approach. Empirical results and discussion are in Section 3. We discuss some related work in Section 4, and conclude in Section 5.

2 Transition Identification under Transition Noise

Biochemical Background A theory of reactions based on collisions requires not only that that the reactant molecules collide with each other, but that they should collide in correct orientations. The reaction:

$$A - B \ + \ C \rightarrow C - A \ + \ B$$

will only occur if C collides with the complex $A-B$ on the "A-side". The probability of such collisions can be mathematically modelled [8]. With small numbers of $A-B$ and C, this reaction may simply not occur. Such non-determinism can be due to extrinsic and intrinsic effects, which has been studied in the chemical

and biological literature [6]. In particular, sources of biological noise are both intrinsic, due to the inherent stochasticity of processes of the system such as gene expression, and extrinsic, due to conditions in the environment. In this paper we focus on intrinsic noise.

Petri nets A Petri net [3] is a bipartite directed graph with two finite sets of nodes, called places and transitions. Arcs are either from a place to a transition, so the place is an input for the transition, or from a transition to a place, in which case the place is an output of the transition. Transitions have a finite number of input places and a finite number of output places. Places can be occupied by zero or more tokens, and arcs can be labelled with an integer weight greater than or equal to one. In a Petri net, a transition is enabled and can hence be executed, or "fired", when the number of tokens at each input place is equal to or greater than the weight on the corresponding arc.

LGTS Transition systems [10] are a formalism used to model the behaviour of dynamic systems. In this paper we adopt the framework of Logical Guarded Transition Systems (LGTS) [16, 17], a generalisation of Petri nets [3] based on logic programming. For example, Fig. 1b shows the "water" Petri net as an LGTS, with the constraint that temperature should be high for the reaction to proceed, which would be encoded as a logic program. Essentially, an LGTS extends the Petri net formalism by allowing logical *guarded* transitions, specifying the pre, post and invariant conditions that must hold for the transition to occur. An LGTS for some system comprises a relation *lgts* from sequences of observed system states to sequences of guarded transitions, plus definitions of guarded transitions, and background knowledge encoding domain-specific and generic constraints. In this paper we will assume that states are as in Petri nets, i.e., vectors of place-values [3]. For transitions to be enabled and executed we additionally require the *guard* of transitions to evaluate to $TRUE$. A guard of a transition is defined in terms of the *pre-* and *post-condition* of the transition and it's *invariant*. The guard evaluates to TRUE if the pre- and post-condition and the invariant are all $TRUE$. Petri nets (and their extended form, that allows "read" and "inhibit" arcs, and variations like coloured and stochastic Petri nets) can be seen as special cases of LGTS's. LGTS's can be used to simulate system behaviour: given some initial state, output a sequence of transitions and their corresponding states. LGTS's can also be identified from samples of system behaviour: given a sequence of states, determine a corresponding sequence of transitions.

Transition identification The identification of LGTS's can be formulated as logical consequence finding [17]. The basic system identification task is as follows:

Given: (a) A sequence S of states, representing observations of the system behaviour; and

(b) Background knowledge B containing generic and domain-specific constraints and definitions of guarded transitions; and

(c) The definition of a relation $G = lgts(S, T)$ that is TRUE for all pairs S and T s.t. T is an LGTS model of S.

Find: An acceptable LGTS. Any LGTS T satisfying $B \wedge G \Rightarrow \exists T\ lgts(S, T)$ is acceptable.

If B and G can be encoded as logic programs, then the T's can be computed using the usual theorem prover used by logic programming systems. With a bound on the number of tokens allowed in each place, the LGTS models for an observation sequence S can be computed by a non-deterministic finite automaton (NFA) [17]. The NFA is a transducer that reads zero or one input symbols (observations) and writes out the corresponding transition.

Transition noise We can distinguish three kinds of "noise" that can affect the identification of transition-based models from data: (a) signal noise (for example, there is an error in the concentration of a metabolite); (b) state noise (for example, a gene is incorrectly recorded as being "on" when it is actually "off"); and (c) transition noise (for example, a reactant is not produced when it normally should be). In principle, both (b) and (c) can be modelled by a Markov process, if the corresponding transition functions were known, which is the assumption made in this paper.

Specifically, we model *transition noise* as a probabilistic transition system. A transition system [10] is a pair $(S, \rightarrow)$ where S is set of states and $\rightarrow$ is a binary relation on S modelling the set of transitions. In a *deterministic* transition system, $\rightarrow$ is a function, i.e., for a given state s there is at most one successor state s' such that $s \rightarrow s'$. In a *non-deterministic* transition system states may have more than one successor state, i.e., $\rightarrow$: $S \times \mathcal{P}(S)$. To model *probabilistic* transitions a probability distribution is defined on each non-deterministic transition.

Identification in the presence of transition noise

Viewing an LGTS as an NFA is useful when considering identification from noisy data. The presence of noise means that there is some mismatch between transitions in background knowledge and the observations. This incompleteness has two aspects: first, there may be some missing intermediate states; second, states may have incorrect values with respect to the corresponding transitions. The first problem is solved based on [4] by generating values for missing place vectors during LGTS identification; in terms of NFA execution this is equivalent to allowing the "empty" input ϵ to correspond to an output transition. The second is solved by allowing the abduction of transitions for "noisy" state pairs; this is akin to employing a theorem-prover that uses SOLD-resolution [18].

These identification steps for noisy observational data will typically lead to an expanded set of transitions. To determine which transitions are more likely

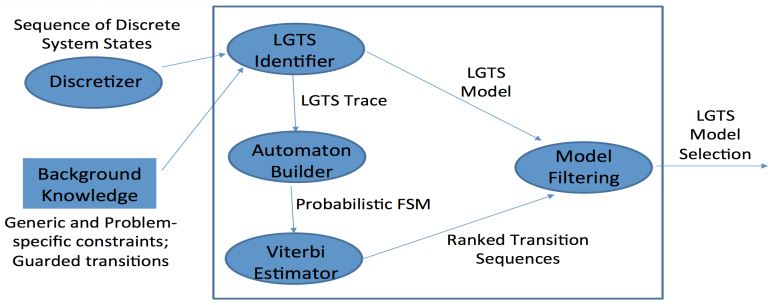


Fig. 2: System identification with noisy data.

to model the underlying system, and which are simply due to noise, the final step in our approach is to sample repeatedly from the distribution over noisy transitions and construct a probabilistic automaton. The resulting probabilities can be used to rank output LGTS models for user inspection, and to evaluate the method.

3 Empirical Evaluation

Our goal is to investigate the identification of the transitions comprising a system, given noisy data sequences representing system behaviour. Specifically: we intend to investigate if the transitions involved in generating the ideal sequence of states can be identified given sufficient numbers of noisy data sequences.

3.1 Problems

The investigation considered system identification for the following problems, listing in order of increasing model-size:

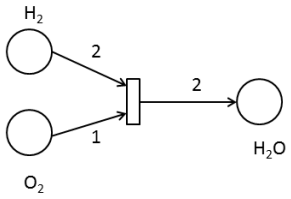


Fig. 3: Network model for the formation of water.

Water. The well-known school-level problem of the formation of water from hydrogen and oxygen forms the simplest system we will examine. The problem clearly consists of a single reaction involving 3 kinds of molecules (places).

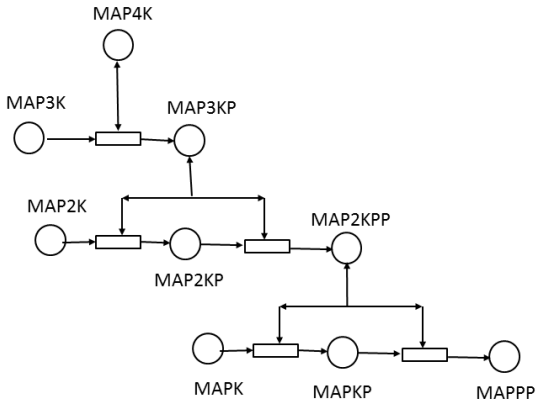


Fig. 4: Network model of the MAPK cascade.

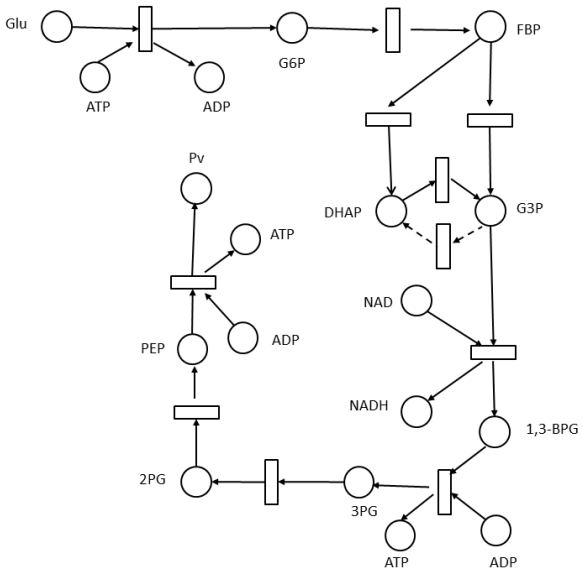


Fig. 5: Network model of the glycolysis pathway. The conversion of DHAP to G3P is taken to be in one-direction only (the reverse is shown by a dashed line, and not identified).

MAPK. The MAPK pathway is a protein-based sequence of events that translate a signal at the cell-surface to the nucleus. The pathway commences when a protein or a hormone binds to a receptor protein that is usually bound to the cell-membrane. This triggers a sequence of events that stops with the DNA expressing one or more genes that alter cell function. At any one step of the cascade, phosphor groups are attached to proteins. This phosphorylated form of the protein then forms a "switch" for commencing the next step. MAPK is a central signalling pathway that is used in all cell-tissues to communicate extra-cellular events to the cell nucleus. It is used to regulate a variety of responses, like hormone action, cell-cycle progression and cell-differentiation. It is also of immense clinical value, since a defect in the pathway often leads to uncontrolled growth. Proteins in the pathway are thus natural targets for anti-cancer drugs.

Glycolysis. The glycolysis pathway was the first metabolic pathway to be discovered. It is a classic case of a series of metabolic reactions in which products of one reaction form the substrates (reactants) for the next reaction. The glycolysis pathway is comprised of 10 such reactions. The reactions breakdown (metabolize) each molecule of glucose into two molecules of pyruvate. The sequence proceeds in three stages: primary (3 reactions), splitting (2 reactions) and phosphorylation (5 reactions). Altogether, 15 metabolites are involved. The pathway is one of the central metabolic pathways in living organisms: it provides an essential part of the energy required for the functioning of a cell, and is used in several metabolic processes.

We note that although Glycolysis contains more metabolites and reactions than MAPK, the latter requires an extended Petri net [3] for its representation (it requires "read" arcs), while Glycolysis (and Water) can be represented by normal Petri nets.

3.2 Data

In this paper, data will be taken to be the result of one or more experiments, each resulting in a table of the following form:

Places	States					
	s_0	s_1	s_2	...	...	s_k
p_1	$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	...	...	$s_{1,k}$
p_2	$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	...	...	$s_{2,k}$
...	...	...	...	...	...	...
...	...	...	...	...	...	...
p_l	$s_{l,0}$	$s_{l,1}$	$s_{l,2}$	...	...	$s_{l,k}$

Places are, as noted above, as in the literature on Petri nets. In experiments in this paper they are restricted to Boolean values (with 0 denoting that the quantity represented by a place is absent, and 1 denoting that it is present in a

sufficient quantity). Each table (experimental result) of this form gives rise to a sequence $(s_0, s_1, \ldots, s_n)$. A noisy data sequence will contain a sequence of states that will differ from an ideal (noise-free) sequence of states.

System behaviours are thus sequences of system states of the form $S_i = (s_{i,0}, s_{i,1}, \ldots, s_{i,n_i})$. Each such sequence can be taken as a set of state-pairs $\{(s_{i,0}, s_{i,1}), (s_{i,1}, s_{i,2}), \ldots, (s_{i,n_i-1}, s_{i,n_i})\}$; and a set of sequences $S = \{S_1, S_2, \ldots, S_j\}$ can be represented by the union of the corresponding sets of state-pairs. We will call this set $\mathit{StatePairs}(S)$.

3.3 Models

An LGTS trace for a state-pair (s_i, s_f) is a set $\mathit{Trace}(s_i, s_f) = \{T_1, T_2, \ldots, T_k\}$, where $T_1 = (t_1, r_1, m_0, m_1), T_2 = (t_2, r_2, m_1, m_2), \ldots, T_k = (t_k, r_k, m_{k-1}, m_k)$, where: (a) each t_j is a guarded transition; (b) $r_j = m_j - m_{j-1}$; and (c) $s_i = m_0$; and (d) $s_f = m_k$. That is, $m_1, m_2, \ldots m_{k-1}$ are intermediate states.

An LGTS model for a state-pair (s_i, s_f) is $T(s_i, s_f) = \{(t, r) : (t, r, m_a, m_b) \in \mathit{Trace}(s_i, s_f)\}$. It is straightforward to extend this to a set of sequences and state-pairs. Given a set of sequences $S = \{S_1, S_2, \ldots, S_j\}$, let $\mathit{TracePairs}(S) = \bigcup_{(s_i,s_j)\in \mathit{StatePairs}(S)} \mathit{Trace}(s_i, s_j)$. Then $\mathit{LGTS}(S) = \{(t, r) : (t, r, m_a, m_b) \in \mathit{TracePairs}(S)\}$.

It is possible to construct a non-deterministic finite-state automaton (NFA) from $\mathit{TracePairs}(S)$ that can output all the sequences in S. Further, a probabilistic finite-state automaton (PFA) can be constructed from the NFA by extending the transition function to include a probability distribution. We omit proofs of these claims here, and show some example automata instead (see Fig. 8 and Fig. 9).

3.4 Algorithms and Machines

Simulated data and probability estimates of the sequences of transitions are obtained using the probabilistic environment provided within the PRISM system [15]. LGTS models are obtained using a Prolog program that implements the basic system identification task in Section 2. All programs were run on a Lenovo dual processor Core i7 laptop, running a Linux emulation with 4 GB of memory.

3.5 Method

Our method is straightforward:

1. Repeat R times:
   For low, medium and high noise levels:
   i. Generate N noisy data sequences, using probabilistic versions of transitions in the ideal model
   ii. Obtain LGTS proofs for each (noisy) data sequence

iii. Using the transition sequences in the LGTS proofs, determine the extent to which the correct transitions can be identified. This may involve an abduction step, somewhat related to SOLD-resolution [18].

The following details are relevant:

1. Low, medium and high levels of noise are defined in terms of the probability with which a probabilistic transition generates the output-state of the corresponding deterministic transition. Here, low noise means that this probability is 90%; medium noise means that the probability is 75%; and high noise means that the probability is 50%.
2. In this paper, $R = 10$. We generate $N = 100$ noisy data sequences for each model. The data are generated using a Hidden Markov (HMM) model, in which states are observed and transitions are hidden.[4] Mainly, two kinds of probabilities have to be specified for the HMM: the conditional probability of emitting a (biological system) state, given a (biological system) transition; and the conditional probability of a transition, given a transition. One additional probability distribution is needed that allows an initial transition to be selected randomly. With each deterministic transition in the model t with input state s_i and output state s_j we associate a probabilistic variant with input s_i and a set S_j as output, with $s_j \in S_j$. The probability with which the HMM emits elements of S_j is determined by the noise level (thus, with low noise, the probability that s_j is emitted is 0.9 and so on). Here, the set S_j consists of s_j and all states within a 1-bit Hamming distance of s_j. The transitions that can follow t with output state s_j are all transitions that have s_j as an input state. The HMM selects amongst these uniformly. This simulation is done using the PRISM program.
3. The efficacy of transition identification is computed as follows. Let the target model of the system consist of a set of transitions T_{act}. Let the system model consist of the set of transitions T_{pred}. We only seek Viterbi probabilities of such sequences of length $|T_{act}|$ (this is provided as a length-bound on the sequences considered by PRISM). The NFA has no loops, and thus $|T_{pred}| = |T_{act}|$. For each experimental run, we compute $E = |T_{act} - T_{pred}|/|T_{act}|$, which is, in effect a false-negative rate. Since the sets T_{act} and T_{pred} are of the same size, the false-positive rate is equal to the false-negative rate. $E = 0$ denotes perfect identification and $E = 1$ denotes perfect mis-identification.

3.6 Results

Table 1 shows some supplementary details related to the transition-identification problem.

Some of the success see in the results of Table 3 may be attributed to the amount of data provided: the results in Table 3 are from 100 (simulated) sequences of observed values. In practice, each such sequence can be thought of as

[4] This may be slightly confusing in the first instance, since in HMMs, states are hidden. Here we are referring to biological system states and biological system transitions.

Noise	Water		MAPK		Glycolysis	
	$\|T\|$	$\|T^*\|$	$\|T\|$	$\|T^*\|$	$\|T\|$	$\|T^*\|$
Low	16 (1)	2 (0)	101 (12)	6 (0)	255 (48)	11 (0)
Med	19 (2)	2 (0)	160 (8)	6 (0)	425 (23)	11 (0)
High	22 (1)	2 (0)	245 (10)	6 (0)	625 (75)	11 (0)

Table 1: Number of transitions (T) in the logical LGTS model and the number of transitions (T^*) identified using the probabilistic model (the latter includes an initial dummy transition). The quantities in parentheses are the standard deviations obtained from multiple repeats of the identification task. All numbers are rounded up to the nearest integer, since fractional transitions are meaningless.

Noise	Error E)		
	Water	MAPK	Glyc.
Low	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Med	0.00 (0.00)	0.03 (0.10)	0.20 (0.10)
High	0.55 (0.35)	0.88 (0.20)	0.88 (0.10)

Table 2: Identification of transitions with small datasets. Here 10 observation sequences are used to identify the transitions.

an experiment; and 100 experiments is unusual in Biology, unless dealing with some form of high-throughput automation like microarray data generation. Table 2 shows the results obtained with significantly fewer observation sequences (10, instead of 100). Now, identification becomes more difficult at moderate noise-levels.

These results suggest that when small amounts of data are the norm, we can expect probabilistic transition-identification to work best at low levels of transition-noise. While this is perhaps obvious enough, a caveat is nevertheless worth noting. A good case can be made that noise-levels that we have labelled here as "medium" and "high" are unlikely to be encountered in practice (for example, a chemical reaction is very unlikely to result in unexpected products 50% of the time). We would therefore expect the transition-identification approach proposed here to work well in practice, even if the data instances are few in number.

Results related to identifying transitions in the target models are in Table 3. The results show clearly that identification of the set of target transitions is perfect at low and medium noise-levels, and only degrades when transition noise is high. It is also apparent that performance is degraded by higher levels of noise, evident from the decrease in Viterbi probability. It is surprising that identification performance is so good, because although the target models studied are relatively simple, with Water, MAPK and Glycolysis having 1, 5 and 10 tran-

Noise	Error E			Probability P		
	Water	MAPK	Glyc.	Water	MAPK	Glyc.
Low	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.81 (0.05)	0.55 (0.03)	0.31 (0.04)
Med	0.00 (0.00)	0.00 (0.0)	0.00 (0.00)	0.55 (0.05)	0.17 (0.04)	0.04 (0.01)
High	0.00 (0.00)	0.20 (0.00)	0.90 (0.10)	0.23 (0.04)	0.01 (0.00)	0.01 (0.00)

Table 3: Identification of transitions in the target model. E denotes the average false-negative rate (that is, the fraction of true transitions not identified). Both the predicted and actual transition sequences are of the same length (see text for details), so the false-negative and false-positive rates are the same. Thus a value of 0.0 for E denotes all–and only–the transitions in the model are identified; and a value of 1.0 denotes none of the transitions in the model are identified. The column P denotes the average Viterbi probability of the highest ranking transition sequence (transitions in this sequence are predicted as being in the target model). Noise levels of low, medium, and high refers to transitions having a probability of 10%, 25% and 50% of resulting in an incorrect state. The quantities in parentheses are the standard deviations obtained from multiple repeats of the identification task.

sitions with 3, 9 and 15 places, respectively, the hypothesis space of *possible* transitions is quite large in each case.

3.7 Transition identification worked example: water

Shown in Figs. 6–9 are examples of the main stages in our approach for the "Water" transition system. In Fig. 6 we see three sample observational state sequences. These are value vectors for the named places over the sequence. In Fig. 7 we show inferred *transitions* in LGTS trace pairs for these sequences, comprising the transition name, difference (reaction) vector, and the corresponding predecessor and successor states for these transitions. Figs. 8 and 9 show the constructed NFA and PFA, respectively, for this system.

The transitions in the highest ranked transition sequence from the PFA (ranked by Viterbi probability obtained using PRISM built-in predicates) are used to identify the system transitions. In Fig. 9, the highest ranked sequence is $(t1, t2)$, and the system model is taken to be $\{(t1, r1), (t2, r2) : (t1, r1), (t2, r2) \in LGTS(S)\}$. Note that this will usually be a subset of $LGTS(S)$ and in some sense, can be considered a generalisation of that set.

4 Related Work

There have been several approaches to the identification of Petri nets from data. For example, in the area of business process mining (BPM) [1] a number of methods construct Petri nets. An important problem in BPM is process discovery, i.e.,

Data S:

$([(h2,0),(o2,0),(h2o,0)],[(h2,1),(o2,1),(h2o,0)],[(h2,0),(o2,0),(h2o,1)])$
$([(h2,0),(o2,0),(h2o,0)],[(h2,1),(o2,1),(h2o,0)],[(h2,0),(o2,0),(h2o,1)])$
$([(h2,0),(o2,0),(h2o,0)],[(h2,1),(o2,1),(h2o,0)],[(h2,1),(o2,0),(h2o,1)])$
...
...

Fig. 6: Transition identification worked example: Noisy data sequences simulating formation of water from hydrogen and oxygen; states shown as place-value tuples.

Transition $TracePairs(S)$:

$(t1,[(h2,1),(o2,1),(h2o,0)],[(h2,0),(o2,0),(h2o,0)],[(h2,1),(o2,1),(h2o,0)])$
$(t2,[(h2,-1),(o2,-1),(h2o,1)],[(h2,1),(o2,1),(h2o,0)],[(h2,0),(o2,0),(h2o,1)])$
...
...

Fig. 7: Transition identification worked example: LGTS trace for data of Fig. 6.

given a log of activity sequences from some software system, generate a model of the business processes involved. This model is often a Petri net, generated, for example, by first extracting a transition system from the activity log and then synthesising a Petri net from it [11]. Although this method is well-founded with theoretical guarantees [5] it has exponential worst-case time complexity and in practice it can lead to both under- and overfitting, so most implementations use additional heuristics [1]. Nonetheless, it has a natural declarative formulation [11] so could provide a basis for further work in ILP.

Another problem in BPM is that of conformance checking, which has been investigated in ILP [12] by supervised learning of integrity constraints to detect non-compliance in activity logs. This was extended to a probabilistic framework using Markov logic and shown to increase accuracy and efficiency [2]. However, these approaches require the use of negative example traces, which are typically not available in biological settings. In [7] an unsupervised ILP approach to BPM was used, but did not use a formalised probabilistic model. In one sense the above BPM tasks are easier than the standard biological setting, since log files typically contain many repeated instances of activity sequences, thus providing enough data to disambiguate Petri net models. In biological settings this is usually not the case due to the difficulty of running many repeated biological experiments.

One heuristic approach often adopted in BPM has been genetic algorithms, and a number of authors working on Petri net identification in biology have also investigated such methods. In [14] grammatical evolution was used, where the

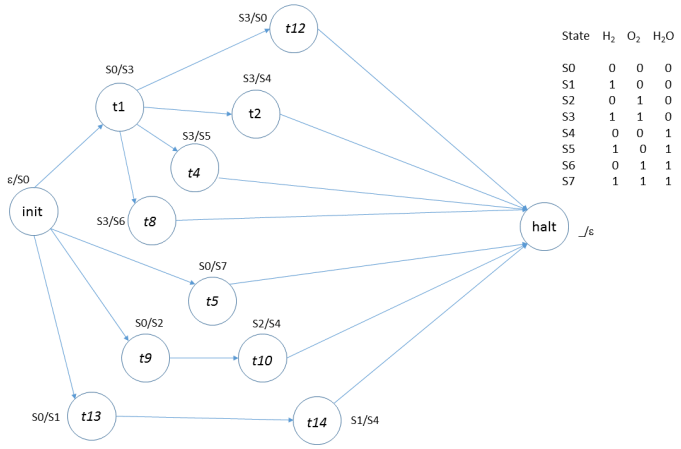


Fig. 8: Transition identification worked example: Non-deterministic finite-state automaton obtained from LGTS trace that correctly derives all data. Transition-identifiers in italics are result of abduction steps made by the theorem-prover to obtain the LGTS trace. Sx/Sy denotes input/ output symbols of transition. All states are connected to "halt" state" — most omitted for clarity.

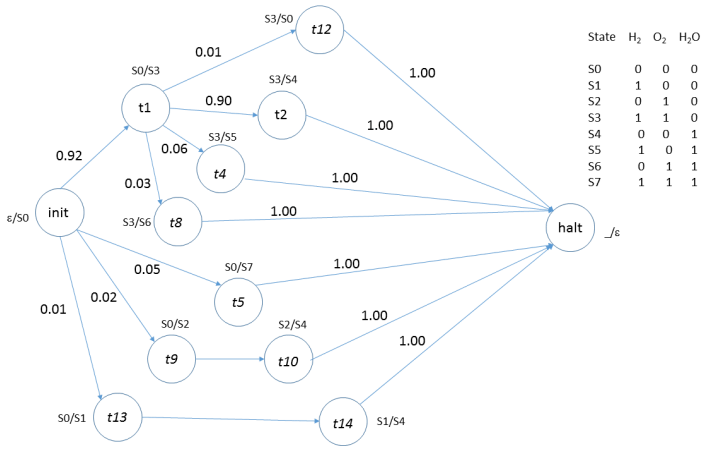


Fig. 9: Transition identification worked example: Probabilistic finite-state automaton obtained from the NFA in (c). Probabilities are estimated using transition sequences followed by the NFA when emitting data sequences as output.

hypothesis space of Petri net models for genetic interactions related to disease was specified using a context-free grammar, and in [13] the incidence matrix was evolved directly. However, these approaches were only reported to work on networks of up to four genes. Furthermore, the above methods cannot identify extended Petri nets, which contain read and inhibitory arcs [17].

The work of [4] appears to be the first method to reconstruct extended Petri nets from time series data. In contrast to our work this approach does not enable constraints on individual transitions. Although there is a form of abduction in this approach, it does not allow for *inductive* steps in identification, as in [16, 17], or probabilistic identification, as in this paper. Learning from interpretation transitions was presented in [9] where an ILP-based approach is used to identify Boolean network models. However, this does not identify probabilistic transitions.

5 Conclusion

We have studied the identification of transition models of biological systems under conditions of added transition noise, extending our previous work. Using the probabilistic logic programming system PRISM [15] we have modelled varying levels of transition noise in three benchmark biological systems. We apply a two-step method, first using a logic-programming approach incorporating both deduction and abduction to identify a complete set of logical guarded transitions that explain the noisy state-sequences. This logical model is then used to construct a probabilistic finite-state automaton. The parameter-estimates obtained (using PRISM) for this automaton are used to identify a subset of the logical model that is then taken to be the system model. Our experimental results show that the method can reconstruct known networks from simulated data with varying amounts of transition-noise.

There are some immediate ways in which the empirical evaluation could be extended. First, we have presented some evidence that as data size decreases, system identification is affected at high noise levels. However, "smaller" data sizes here is still much higher than what is normally available in experimental life-sciences. For example, how will system identification be affected when data are available from 2 to 3 experiments? Secondly, we have seen that larger sized networks are affected more by high noise-levels than networks of smaller size. Although the high noise-levels used here are unlikely to be encountered in real data, there is nevertheless a need for further work to investigate the effect of network size. It is possible that, even at low noise levels, system identification may degrade for very large networks. Third, although we have examined the effect of incorrect data, we have not examined the effect of incomplete data. The probabilistic setting we have used naturally accounts for this by the use of the EM algorithm to estimate probabilities. We should therefore be able to investigate the effect of missing data on system identification.

References

1. W. Van Der Aalst. Process Mining: Overview and Opportunities. *ACM Transactions on Management Information Systems*, 3(2), 2012.
2. E. Bellodi, F. Riguzzi, and E. Lamma. Probabilistic Declarative Process Mining. In Y. Bi and M.-A. Williams, editors, *KSEM 2010*, volume LNAI 6291, pages 292–303. Springer, 2010.
3. R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, Berlin, Second edition, 2010.
4. M. Durzinsky, A. Wagler, and W. Marwan. Reconstruction of extended Petri nets from time series data and its application to signal transduction and to gene regulatory networks. *BMC Systems Biology*, 5:113, 2011.
5. A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures: Part II: State Spaces of Concurrent Systems. *Acta Informatica*, 27:343–368, 1990.
6. M. Elowitz, A. Levine, E. Siggia, and P. Swain. Stochastic Gene Expression in a Single Cell. *Science*, 297:1183–1186, 2002.
7. S. Ferilli and F. Esposito. A Logic Framework for Incremental Learning of Process Models. *Fundamenta Informaticae*, 128:1–31, 2013.
8. D. Gillespie. Exact Stochastic Simulation of Coupled Chemical Reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.
9. K. Inoue, T. Ribeiro, and C. Sakama. Learning from interpretation transition. *Machine Learning*, 94(1):51–79, 2014.
10. R. Keller. Formal Verification of Parallel Programs. *Communications of the ACM*, 19(7):371–384, July 1976.
11. E. Kindler, V. Rubin, and W. Schafer. Process Mining and Petri Net Synthesis. In J. Eder and S. Dustdar, editors, *BPM 2006 Workshops*, volume LNCS 4103, pages 105–116. Springer, 2006.
12. E. Lamma, P. Mello, F. Riguzzi, and S. Storari. Applying Inductive Logic Programming to Process Mining. In H. Blockeel, J. Ramon, J. Shavlik, and P. Tadepalli, editors, *ILP 2007: Proceeding of the International Conference on Inductive Logic Programming*, volume LNAI 4894, pages 132–146. Springer, 2008.
13. M. Mayo. Learning Petri Net Models of Non-Linear Gene Interactions. *Biosystems*, 82(1):74–82, 2005.
14. J. Moore, E. Boczko, and M. Summar. Connecting the dots between genes, biochemistry, and disease susceptibility: systems biology modeling in human genetics. *Molecular Genetics and Metabolism*, 84:104–111, 2005.
15. T. Sato and Y. Kameya. PRISM: A symbolic-statistical modeling language. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI97)*, pages 1330–1335, 1997.
16. A. Srinivasan and M. Bain. Knowledge-Guided Identification of Petri Net Models of Large Biological Systems. In S. Muggleton, A. Tamaddoni-Nezhad, and F. Lisi, editors, *Proc. 21st Intl. Conference on Inductive Logic Programming (ILP 2011; Revised Selected Papers)*, volume 7207 of *Lecture Notes in Computer Science*, pages 317–331, Berlin, 2012. Springer.
17. A. Srinivasan and M. Bain. Identification of Transition-Based Models of Biological Systems using Logic Programming. Technical Report UNSW-CSE-TR-201425, School of Computer Science and Engineering, University of New South Wales, Sydney, Australia, December 2014.
18. A. Yamamoto. Representing Inductive Inference with SOLD-Resolution. In *Proceedings of the IJCAI'97 Workshop on Abduction and Induction in AI*, 1997.