BIOINFORMATICS ORIGINAL PAPER

Vol. 23 no. 3 2007, pages 336–343
doi:10.1093/bioinformatics/btl596

Systems biology

Qualitatively modelling and analysing genetic regulatory networks: a Petri net approach

L. Jason Steggles*, Richard Banks, Oliver Shaw and Anil Wipat
School of Computing Science, University of Newcastle, Newcastle upon Tyne, UK

Received on June 16, 2006; revised on November 6, 2006; accepted on November 18, 2006
Advance Access publication November 22, 2006
Associate Editor: Satoru Miyano

Downloaded from https://academic.oup.com/bioinformatics/article/23/3/336/235741 by guest on 16 August 2022

ABSTRACT

Motivation: New developments in post-genomic technology now provide researchers with the data necessary to study regulatory processes in a holistic fashion at multiple levels of biological organization. One of the major challenges for the biologist is to integrate and interpret these vast data resources to gain a greater understanding of the structure and function of the molecular processes that mediate adaptive and cell cycle driven changes in gene expression. In order to achieve this biologists require new tools and techniques to allow pathway related data to be modelled and analysed as network structures, providing valuable insights which can then be validated and investigated in the laboratory.

Results: We propose a new technique for constructing and analysing qualitative models of genetic regulatory networks based on the Petri net formalism. We take as our starting point the Boolean network approach of treating genes as binary switches and develop a new Petri net model which uses logic minimization to automate the construction of compact qualitative models. Our approach addresses the shortcomings of Boolean networks by providing access to the wide range of existing Petri net analysis techniques and by using non–determinism to cope with incomplete and inconsistent data. The ideas we present are illustrated by a case study in which the genetic regulatory network controlling sporulation in the bacterium *Bacillus subtilis* is modelled and analysed.

Availability: The Petri net model construction tool and the data files for the *B. subtilis* sporulation case study are available at http://bioinf.ncl.ac.uk/gnapn

Contact: L.J.Steggles@ncl.ac.uk

1 INTRODUCTION

New developments in post-genomic technology now provide researchers with the data necessary to study regulatory processes in a holistic fashion at multiple levels of biological organization (Spellman *et al*., 1998). One of the major challenges for the biologist is to integrate and interpret these vast data resources to gain a greater understanding of the structure and function of the molecular processes that mediate adaptive and cell cycle driven changes in gene expression. The visualization and computational representation of these complex regulatory processes as network structures facilitates the application of a range of analysis and simulation techniques that can, in turn, shed light on our understanding of their organization and behaviour (Uetz *et al*., 2000). The knowledge gained from these analyses should then provide valuable insights, allowing the formulation of hypotheses which can then be tested in the laboratory. In order to achieve this vision, biologists require new automated formal techniques to allow pathway related data to be modelled and analysed.

In this paper, we present a new technique for qualitatively modelling and analysing genetic regulatory networks. We take as our starting point Boolean networks (Kauffman, 1969; Akutsu *et al*., 1999), an existing modelling approach for regulatory networks in which regulatory entities (i.e. genes, proteins and external signals) are viewed abstractly as binary switches. While Boolean networks have proved successful in modelling real world genetic regulatory networks (Huang, 1999; Szallasi and Liang, 1998), they suffer from a number of shortcomings: analysis can be problematic due to the exponential growth in Boolean states and the lack of tool support; and they do not cope well with the inconsistent and incomplete data that often occurs in practice. To address these problems, we propose a new model for genetic regulatory networks based on Petri nets (Reisig, 1985; Murata, 1989), a well developed formal framework for modelling and analysing complex concurrent systems (Reisig and Rozenberg, 1998). We illustrate our modelling approach by presenting a case study in which part of the genetic regulatory network for initiating sporulation in *Bacillus subtilis* (Stragier and Losick, 1996; Stephenson and Lewis, 2005) is modelled and analysed.

A range of initial investigations into using Petri nets to model biological systems have appeared in the literature to date, including: place/transition nets (Reddy *et al*., 1996; Chaouiya *et al*., 2004; Simão *et al*., 2005); stochastic nets (Gross and Peccoud, 1998; Tsavachidou and Leibman, 2002; Shaw *et al*., 2006); high–level nets (Heiner *et al*., 2001; Comet *et al*., 2005); and hybrid nets (Matsuno *et al*., 2000). An interesting comparison of these different approaches is presented in Peleg *et al*. (2005). The results we present significantly extend the related ideas presented in Chaouiya *et al*. (2004), both semantically and in the provision of automated tool support for model construction and analysis.

This paper is organized as follows. In Section 2 we give a brief introduction to the theory of Petri nets. In Section 3 we describe a new approach to modelling the Boolean behaviour of genetic regulatory networks using Petri nets. In Section 4 we consider a case study in which we apply our techniques to modelling and analysing the genetic regulatory network for sporulation in *B.subtilis*. Finally, in Section 5, we present some concluding remarks.

Note in the sequel we assume the reader is familiar with the basic Boolean operators *not*, *or* and *and* (Grossman, 2002).

*To whom correspondence should be addressed.

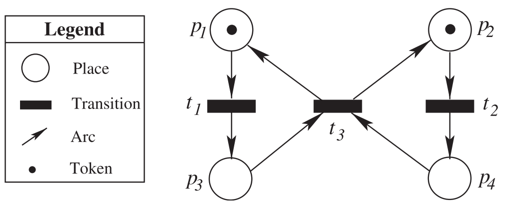


Fig. 1. A simple example of a Petri net.

2 PETRI NETS

The theory of Petri nets (Reisig, 1985; Murata, 1989) provides a graphical notation with a formal mathematical semantics for modelling and reasoning about concurrent, distributed systems. A Petri net is a directed bipartite graph and consists of four basic components: *places*, which are denoted by circles; *transitions* denoted by black rectangles; *arcs* denoted by arrows; and *tokens* denoted by black dots. A simple example of a Petri net is given in Figure 1. The places, transitions and arcs describe the static structure of the Petri net. Each transition has a number of *input places* (places with an arc leading to the transition) and a number of *output places* (places with an arc leading to them from the transition). We normally view places as representing resources or conditions and transitions as representing actions or events (Murata, 1989). Note arcs that directly connect two transitions or two places are not allowed.

The state of a Petri net is given by the distribution of tokens on places within it, referred to as a *marking*. The *state space* of a Petri net is therefore the set of all possible markings. The dynamic properties of the system are modelled by transitions which can fire to move tokens around the places in a Petri net. Transitions are said to be *enabled* if each of their input places contain at least one token. An enabled transition can *fire* by consuming one token from each of its input places and then depositing one token on each of its output places. For example, in Figure 1 both transitions t_1 and t_2 are enabled. Firing transition t_1 would result in a token being taken from place p_1 and a new token being deposited on place p_3. Note that the firing rule is normally generalized to allow arcs to contain weights indicating the number of tokens to be consumed or produced when a transition is fired (Murata, 1989). Since such arc weights are not required in the sequel we have omitted them from our introduction for simplicity.

At any given instance a number of transitions may be enabled to fire (as in the example in Fig. 1). In such a case, a transition is chosen non–deterministically to fire. A marking m_2 is said to be reachable from a marking m_1 if there is a sequence of transitions that can be fired starting from m_1 which results in the marking m_2. A Petri net is said to be *k–bounded* if in all reachable markings no place has more than k tokens. A Petri net which is 1–bounded is said to be *safe*. Safeness is an important property since any safe Petri net is well–suited to automatic analysis (Reisig and Rozenberg, 1998).

An important advantage of Petri nets is that they are supported by a wide range of theoretically well–founded techniques and tools for simulation and analysis. For example, Petri nets can be automatically checked for boundedness and the presence of deadlocks (markings in which no transitions are enabled to fire) (Reisig and Rozenberg, 1998). A Petri net can also be analysed by constructing its *reachability graph* (Murata, 1989) which captures the possible firing sequences that can occur from a given initial marking. A range of techniques based on model checking (Esparza, 1994; Khomenko, 2003) have been developed for analysing reachability properties and these provide a means of coping with the potentially large state space of a Petri net model.

3 MODELLING APPROACH

In this section we present a new qualitative model for genetic regulatory networks based on Petri nets. The idea is to use an intuitive Petri net structure to represent the Boolean relationships that exisit between regulatory entities. We start by defining each entity's individual behaviour as a truth-table following the approach used in *Boolean networks* (Akutsu *et al*., 1999). We then extract from these truth-tables the fundamental relationships between entities as Boolean terms by applying logic minimization techniques (Grossman, 2002; Breeding, 1992). In particular, we automate this process using the *Espresso* (Brayton *et al*., 1984) logic minimization tool. We then directly translate these Boolean terms into appropriate Petri net control structures to produce a compact Petri net model that correctly captures the original Boolean behaviour of a genetic regulatory network. Both the synchronous and asynchronous semantic interpretation of Boolean networks (Gershenson, 2002) can be modelled using our approach. We choose to focus on the synchronous semantics here and develop a simple two phase commit protocol to allow synchronized state updates within the asynchronous Petri net framework. Once constructed, our Petri net models can be simulated and analysed using the wide range of existing Petri net tools. These tools provide a range of powerful, theoretically well–founded analysis techniques (Reisig and Rozenberg, 1998; Esparza, 1994) which allow researchers to investigate the structure and dynamic behaviour of a genetic regulatory network.

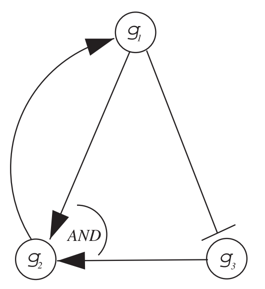
3.1 Boolean networks

In a Boolean network (Kauffman, 1969; Akutsu *et al*., 1999) the state of each regulatory entity is represented as a Boolean value, either 1 representing the entity is active (e.g. a gene is expressed or a protein is present) or 0 representing the entity is inactive (e.g. a gene is not expressed or a protein is absent). The state of a genetic regulatory network containing n entities is then naturally represented as a Boolean vector $[g_1, \ldots, g_n]$ and this gives us a state space containing 2^n states (Akutsu *et al*., 1999). The behaviour of each g_i is described using a Boolean function f_i which, given the current states of the entities in its neighbourhood (i.e. those entities which directly affect it), defines the next state for g_i. As an example consider the Boolean network in Figure 2a (Akutsu *et al*., 1999) which contains three entities g_1, g_2 and g_3. The next state g'_i of each entity is defined by the truth-table in Figure 2b or equivalently by the following Boolean functions:

$$g'_1 = g_2,\ g'_2 = g_1 g_3,\ g'_3 = \overline{g}_1$$

where the notation $\bar{x}$, $x + y$ and xy is used to represent the Boolean operators not, or and and (Grossman, 2002), respectively. The dynamic behaviour of a Boolean network can be semantically interpreted either asynchronously where genes update their state independently, or synchronously where all genes update their state together (see Gershenson, 2002). We focus on the synchronous semantics in this paper which appears to be widely used in the literature (Bower and Bolouri, 2001; Gershenson, 2002). The synchronous behaviour for our example Boolean network is shown in the

Downloaded from https://academic.oup.com/bioinformatics/article/23/3/336/235741 by guest on 16 August 2022



(a) Boolean network

Current			Next		
g_1	g_2	g_3	g_1'	g_2'	g_3'
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

(b) Truth table

Fig. 2. An example of a Boolean network for three entities g_1, g_2 and g_3.

truth-table in Figure 2b, where each row represents a synchronized state transition.

Boolean networks have proved successful in modelling real world regulatory networks (Huang, 1999; Szallasi and Liang, 1998). However, their application in practice is hindered by a number of shortcomings. In particular, analysis can be problematic due to the exponential growth in Boolean states and the lack of tool support in this area. They are also unable to cope with the inconsistent and incomplete regulatory network data that often occurs in practice. For this reason we consider extending the Boolean network approach by developing a Petri net based Boolean model.

3.2 Deriving Boolean regulatory relationships

Given a set of truth-tables defining the Boolean behaviour of a genetic network we would like to extract a compact representation of the regulatory relationships between entities. We address this using well–known techniques from Boolean logic (Breeding, 1992; Grossman, 2002) which allow us to derive Boolean terms describing the functional behaviour of each entity. The idea is to consider the truth-table for each entity and to list all the states which result in a next state in which the entity is active (i.e. in state 1). For example, in the truth-table given in Figure 2b the entity g_1 is active after the states 010, 011, 110 and 111 (where *xyz* denotes the state $g_1 = x$, $g_2 = y$ and $g_3 = z$). We can represent each state as a product term, called a minterm (Grossman, 2002), using the *and* Boolean operator, where the variable g_i is used to represent that an entity g_i is in state 1, and the negated variable $\overline{g}_i$ represents that an entity g_i is 0. So the state 010 for g_1 is represented by the minterm $\overline{g}_1 g_2 \overline{g}_3$. Applying this approach and then summing the derived minterms using the *or* Boolean operator allows us to derive a Boolean term in disjunctive normal form (Grossman, 2002) that defines the functional behaviour of an entity. Continuing with our example, we derive the following Boolean term for gene g_1:

$$\overline{g}_1 g_2 \overline{g}_3 + \overline{g}_1 g_2 g_3 + g_1 g_2 \overline{g}_3 + g_1 g_2 g_3$$

Note that this term completely defines the functional behaviour of g_1, i.e. whenever the term above evaluates to 1 in a state we know g_1 will be active in the next state, and whenever the term is 0 we know g_1 will be inactive. Using this technique we can construct a Boolean network that completely specifies the functional behaviour of a genetic network.

The Boolean terms derived above are often unnecessarily complex and can normally be simplified using *logic minimization* (Breeding, 1992; Grossman, 2002). From a biological point of view, this simplification process is important as it helps to identify the underlying regulatory relationships that exist between entities in a genetic network. The idea behind logic minimization is to simplify Boolean terms by merging minterms that differ by only one variable. As an example, consider the term $\overline{g}_1 g_2 \overline{g}_3 + \overline{g}_1 g_2 g_3$ which contains two minterms. Note that these two minterms differ by only one variable g_3 and this implies that the value of g_3 is unimportant in determining the value of the term. We can therefore simplify the term by merging the two minterms to produce a simpler term $\overline{g}_1 g_2$ which is logically equivalent (Grossman, 2002). For brevity we omit the full details of Boolean logic minimization here (we refer the interested reader to Breeding, 1992) and instead illustrate the idea by simplifying the Boolean terms in our running example:

$$\begin{aligned}
(g_1')\quad & \overline{g}_1 g_2 \overline{g}_3 + \overline{g}_1 g_2 g_3 + g_1 g_2 \overline{g}_3 + g_1 g_2 g_3 \\
& \Rightarrow \overline{g}_1 g_2 + g_1 g_2 \overline{g}_3 + g_1 g_2 g_3 \\
& \Rightarrow \overline{g}_1 g_2 + g_1 g_2 \Rightarrow g_2, \\
(g_2')\quad & g_1 \overline{g}_2 g_3 + g_1 g_2 g_3 \Rightarrow g_1 g_3, \\
(g_3')\quad & \overline{g}_1 \overline{g}_2 \overline{g}_3 + \overline{g}_1 \overline{g}_2 g_3 + \overline{g}_1 g_2 \overline{g}_3 + \overline{g}_1 g_2 g_3 \\
& \Rightarrow \overline{g}_1 \overline{g}_2 + \overline{g}_1 g_2 \overline{g}_3 + \overline{g}_1 g_2 g_3 \\
& \Rightarrow \overline{g}_1 \overline{g}_2 + \overline{g}_1 g_2 \Rightarrow \overline{g}_1.
\end{aligned}$$

Note that the minimized Boolean terms presented above correctly correspond to the Boolean network definition in Section 3.1.

3.3 Modelling Boolean networks using Petri nets

While the Boolean terms derived in Section 3.2 compactly capture the behaviour of a Boolean network they are not amenable to analysis in their current form. We address this by translating these terms into appropriate Petri net control structures. The approach we take is to represent the Boolean state of each entity g_i in a Petri net by the well–known approach (see for example Reisig, 1985; Chaouiya *et al*., 2004) of using two complementary places Pi and $\overline{Pi}$, where a token on place Pi indicates the entity is active, $g_i = 1$, and a token on place $\overline{Pi}$ that it is not, $g_i = 0$. Note the total number of combined tokens on places Pi and $\overline{Pi}$ will therefore always be equal to 1.

Since Petri nets fire transitions asynchronously it is straightforward to model the asynchronous behaviour of a Boolean network in this setting (see Chaouiya *et al*., 2004 for a related approach). We focus on modelling the synchronous behaviour of a Boolean network here and make use of a two phase commit protocol to synchronise updates in our model. In the first phase of the protocol each entity g_i in the model decides whether it should be active or not in the next state. This decision is recorded using two places, *Pi_On* and *Pi_Off*, where a token on *Pi_On* indicates g_i is active in the next state and a token on *Pi_Off* that it is not. When all the entities have made a decision about their next state the second phase of the protocol begins and the state of each entity is synchronously updated.

Phase one: *update decisions*. We begin by considering under what conditions each entity is active (i.e. in state 1) in its next state and use the process detailed in Section 3.2 to derive a minimized Boolean term which compactly captures these conditions. We then model the minimized Boolean term for each entity g_i in our Petri net by adding a separate transition to represent each minterm it contains. The idea is that each transition will be enabled to fire,

Downloaded from https://academic.oup.com/bioinformatics/article/23/3/336/235741 by guest on 16 August 2022

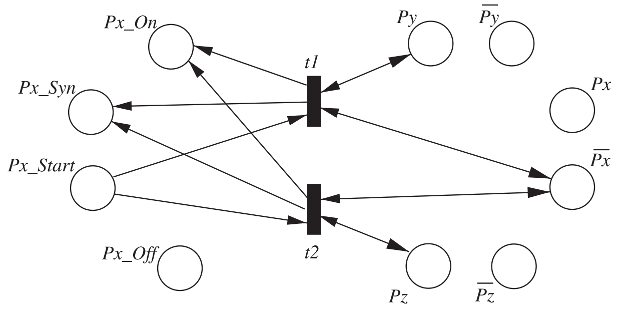


Fig. 3. Transitions modelling the Boolean term $\overline{g}_x\, g_y + \overline{g}_x\, g_z$ for gene g_x.

placing a token on *Pi_On*, precisely when the corresponding minterm is true. As an example, suppose the following minimized Boolean term

$$\overline{g}_x g_y + \overline{g}_x g_z$$

has been derived for a gene g_x. Then the first minterm $\overline{g}_x g_y$ tells us that gene g_x should be expressed in the next state (i.e. $g_x = 1$) when genes $g_x = 0$ and $g_y = 1$ in the current state. We model this minterm using the transition *t*1 depicted in Figure 3. This transition fires when places $\overline{Px}$ and *Py* contain a token (i.e. when $g_x = 0$ and $g_y = 1$) and results in a token being placed on *Px_On* (indicating g_x is expressed in the next state). Note the use of read arcs (Murata, 1989) here, i.e. bidirectional arcs which do not consume tokens but just check they are present. This ensures the tokens on places $\overline{Px}$ and *Py* are not removed at this stage (doing so would corrupt the current state of the genes). The start place *Px_Start* is used as a control input to the transition to ensure only one decision is made for gene g_x during a single protocol step (update transitions for g_x can only fire if a token is present on *Px_Start*). The place *Px_Syn* is used to indicate when an update decision has been made for gene g_x, information needed by the protocol to determine when the first phase is complete. This process is then repeated to model the complete Boolean term for a given entity. In our example, this results in transition *t*2 (Fig. 3) being added to model the minterm $\overline{g}_x g_z$.

It remains to model the complementary decision procedure for deciding when an entity is inactive in the next state, that is when *Pi_Off* should be marked. To do this we simply apply the process detailed above again but this time consider the conditions under which each entity becomes inactive. We then model the resulting Boolean terms as explained above except that this time each transition we add will mark place *Pi_Off* instead of *Pi_On*.

Phase two: *synchronous state update*. After all the entities have made their update decisions all the synchronisation places will be marked and this allows the control transition depicted in Figure 4a to fire, initiating the second phase of the protocol. This phase performs a synchronised update step in which the state of each entity g_i is updated in turn by placing a token on *Pi* if place *Pi_On* is marked or on $\overline{Pi}$ if place *Pi_Off* is marked. An example fragment of the Petri net structure used for this update is given in Figure 4b for an arbitrary entity g_i. The fragment contains four transitions which represent the four possible update situations that can occur: move token from place $\overline{Pi}$ to *Pi*; leave token on *Pi*; move token from place *Pi* to $\overline{Pi}$; leave token on $\overline{Pi}$. Only one of these transitions will be enabled to fire. Once the gene g_i has updated its state a token

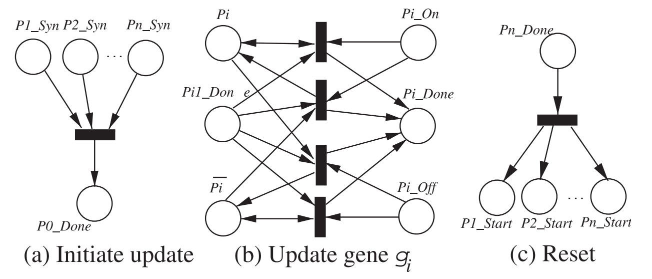


Fig. 4. Petri net fragments for controlling synchronous updates.

Table 1. Truth-table for entity g_2

g_1	g_3	g_2
0	0	0
0	1	0
1	0	?
1	1	1

is placed on place *Pi_Done* to indicate that the next entity can be updated. When the last entity g_n has been updated place *Pn_Done* will be marked and the control transition depicted in Figure 4c initiates a reset step which re-marks the start places, allowing the whole synchronization protocol to begin again.

3.4 Inconsistent and incomplete data

So far we have assumed that we are always able to derive complete and consistent truth-tables which correctly capture the behaviour of each entity in a genetic network. However, in practice it is rarely the case that a genetic network is fully understood and indeed, this is one important reason for modelling such networks. The data provided may be incomplete in the sense that information is missing about what happens in certain states, or it may be inconsistent in that we have conflicting information. The result is that the behaviour of some entities under certain conditions may be unknown.

Such incomplete and/or inconsistent behavioural information is problematic for standard Boolean network models which are unable to represent the possibility of more than one next state. However, Petri nets are a non-deterministic modelling formalism (Reisig, 1985) which are able to represent conflicting choices and unknown behaviour by incorporating all possible next state transitions. The idea is to identify for each entity all the problematic states in which the next state is unknown and then to include these states when deriving Boolean terms using logic minimization for both the active and inactive next state behaviour. The result will be a model in which for any unknown state there will be two conflicting transitions enabled, representing the choice between an active or inactive next state for the associated entity. In this way we allow these problematic states to exhibit both possible behaviours and rely on the non–deterministic choice mechanisms of the Petri net model.

As an example, consider the truth-table defining the behaviour of an entity g_2 presented in Table 1. The question mark indicates that when $g_1 = 1$ and $g_3 = 0$ the next state for g_2 is unknown. Using our approach, we add the minterm $g_1\overline{g}_3$ representing the unknown state to the terms derived to define the active and inactive next state of g_2,

Downloaded from https://academic.oup.com/bioinformatics/article/23/3/336/235741 by guest on 16 August 2022

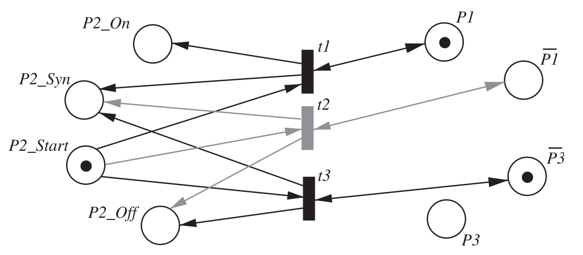


Fig. 5. Transitions modelling the partial truth-table for g_2.

resulting in the following defining equations:

$$g_2 = g_1g_3 + g_1\overline{g}_3,\ \overline{g}_2 = \overline{g}_1\overline{g}_3 + \overline{g}_1g_3 + g_1\overline{g}_3.$$

Applying logic minimization allows us to simplify these to $g_2 = g_1$ and $\overline{g}_2 = \overline{g}_1 + \overline{g}_3$ which will then be modelled using three transitions as shown in Figure 5. Note that transitions *t*1 and *t*3 are in conflict since they are both enabled to fire in state $g_1 = 1$ and $g_3 = 0$ (Fig. 5) and this represents the non–deterministic choice between the two possible next states for g_2 in this unknown state.

We can still perform meaningful analysis on the resulting Petri net models since the Petri net tools are able to cope with such non–deterministic choices. As more data becomes available for a genetic network the Petri net model can be refined to reduce the amount of non-determinism it contains and thus Petri nets provide an interesting means of documenting the development of knowledge about a genetic network.

3.5 Tool support for model construction

To support the modelling approach presented above we have developed a prototype tool to automate the model construction process. The tool takes as input a series of truth-tables describing the behaviour of the entities in a genetic network. These truth-tables are allowed to contain inconsistent and incomplete data as discussed in Section 3.4 above. From these tables the tool is able to automatically extract the minimized Boolean terms using the Espresso logic minimization tool (Brayton *et al*., 1984). It then uses these terms to construct a Petri net model of the given genetic network which can be based on either the synchronous or asynchronous Boolean network semantics (Gershenson, 2002). The resulting Petri net model can then be exported as a PNML (Petri net markup language) (Billington *et al*., 2003) file to a wide range of existing Petri net tools to be simulated and analysed. This prototype Petri net construction tool is freely available for academic use from our project website http://bioinf.ncl.ac.uk/gnapn.

4 CASE STUDY: SPORULATION IN *B.SUBTILIS*

In this section we illustrate our approach by presenting a case study in which part of the genetic regulatory network for initiating sporulation in *B.subtilis* (Stragier and Losick, 1996; Stephenson and Lewis, 2005) is modelled and analysed. Using the detailed data provided in de Jong *et al*. (2004) we define the Boolean behaviour of the key regulatory entities involved in initiating sporulation using truth-tables. We then apply our prototype tool to automatically construct a qualitative Petri net model capturing the behaviour of the sporulation regulatory network. This Petri net is then validated and analysed using PEP (Grahlmann, 1997) and in particular, model checking techniques (Esparza, 1994) are applied to gain a deeper understanding of the model's behaviour.

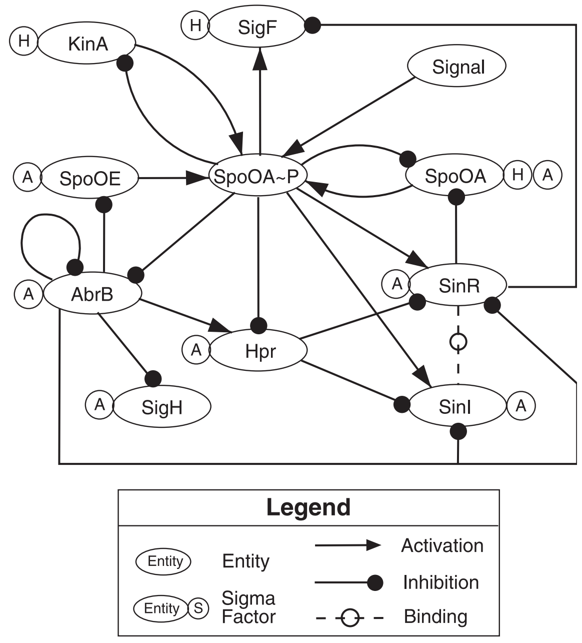


Fig. 6. Genetic regulatory network for sporulation in *B.subtilis*.

4.1 Regulating sporulation in *B.subtilis*

The soil bacterium *B.subtilis* is able to survive extreme conditions by forming dormant spores which are resistant to adverse environmental conditions (Stragier and Losick, 1996). This process of sporulation is controlled by a complex regulatory network, a small part of which is shown abstractly in Figure 6 (adapted from de Jong *et al*., 2004). The presence or absence of adverse environmental conditions, such as nutritional starvation, is represented by a signalling entity Signal; when Signal is active it indicates that the bacteria is under nutritional stress and needs to consider initiating sporulation. At the centre of the regulatory network is the phosphorylation of the protein SpoOA, which in turn activates a cascade of sigma factors which direct the transcription of genes that initiate sporulation. This *phosphorelay* (Stragier and Losick, 1996) transfers a phosphate from the kinase KinA (amongst others) via a number of intermediate steps to activate the protein SpoOA by producing SpoOA$\sim$P. This in turn activates the production of SigF, a key sigma factor whose expression we take as an indication that the sporulation process has been initiated. The phosphatase SpoOE is able to reverse the phosphorylation of SpoOA, thereby inactivating SpoOA and preventing sporulation. The above interactions form part of a complex regulatory control mechanism involving a number of other genes which act as transition state regulators to inhibit or activate the sporulation process for a more detailed account of the sporulation process see (Stragier and Losick, 1996; Stephenson and Lewis, 2005).

4.2 Constructing the Petri net model

The process of constructing a Petri net model for a genetic regulatory network begins by identifying the entities (i.e. genes, proteins or stimuli) we need to consider. For our case study, there

Downloaded from https://academic.oup.com/bioinformatics/article/23/3/336/235741 by guest on 16 August 2022

Table 2. Truth-table for AbrB

SigA	AbrB	SpoOA∼P	AbrB
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

are 12 entities (de Jong *et al*., 2004): SigF, KinA, SpoOA, SpoOA∼P, AbrB, SpoOE, SigH, Hpr, SinR, SinI, SigA and Signal. In our Boolean model we consider each entity as either being in an *on* state (i.e. sufficiently present to have an affect) or an *off* state (i.e. not sufficiently present to have an affect). The last two entities, SigA and Signal, are not regulated within the scope of our network and as such their state remains fixed once it has been initialized. For each of the remaining 10 entities we use the available literature to construct a truth-table to abstractly define how each entity's state is affected by the state of the entities in its neighbourhood. Note it is important that the information sources used are carefully documented so that we are able to clearly see the evidence and assumptions our resulting model is based on.

As an example, consider the entity AbrB. Using de Jong *et al*. (2004) and Figure 6 we can derive that its neighbourhood consists of three entities, namely SigA, AbrB and SpoOA∼P. We can then consider how the state of AbrB is affected by these entities (de Jong *et al*., 2004; Stragier and Losick, 1996; Stephenson and Lewis, 2005) and this results in the truth-table depicted in Table 2.

The next step is to apply logic minimization to the truth-tables we have derived to extract Boolean equations which define when each entity will be active (i.e. in state 1) and inactive (i.e. in state 0). The resulting Boolean equations extracted from the truth-table for AbrB above are presented as equations below:

$$\mathrm{AbrB} = (\mathrm{SigA}\ \overline{\mathrm{AbrB}}\ \overline{\mathrm{SpoOA}\sim\mathrm{P}}),$$
$$\overline{\mathrm{AbrB}} = \overline{\mathrm{SigA}} + \mathrm{AbrB} + \mathrm{SpoOA}\sim\mathrm{P}.$$

Applying the above process to each of the remaining entities in our network results in a set of Boolean equations that define the qualitative behaviour of the sporulation genetic network. These equations are listed in full in the Supplementary information provided online for this paper. This process is automated by our support tool (see Section 3.4) which then completes the process by automatically constructing a Petri net model of the regulatory network (see Section 3.3) which contains 75 places and 91 transitions.

4.3 Model validation and analysis

Once we have constructed our Petri net model we can then validate and analyse it using the wide range of tools available for Petri nets. In this case study we use the PEP tool (see Grahlmann, 1997, and the website http://theoretica.informatik.uni-oldenburg.de/~pep/) to analyse our model and in particular, consider using model checking techniques (Esparza, 1994; Khomenko, 2003). Our aim is to illustrate the range of analysis possible using available Petri net tools, from simple validation tests to more indepth gene knock out and overexpression analysis.

Table 3. Simulation results in the presence of nutritional stress

Entity	States					
SigF	0	0	0	0	0	1
KinA	0	0	0	1	0	0
SpoOA	1	1	1	1	1	0
SpoOA∼P	0	0	0	0	1	0
AbrB	1	0	1	0	1	0
SpoOE	1	0	1	0	1	0
SigH	0	0	1	0	1	0
Hpr	1	1	0	1	0	0
SinR	1	0	0	0	0	0
SinI	0	0	0	0	0	0

4.3.1 Model validation We begin the analysis process by validating our model to ensure it is a reasonable representation of the regulatory system in question. The idea is to perform a range of tests on the model to ensure that it satisfies the basic behavioural properties indicated by the experimental results in the literature. In the case of the sporulation regulatory network the main control structure is the phosphorelay (de Jong *et al*., 2004; Stragier and Losick, 1996), which is responsible for the phosporylation of SpoOA under nutritional stress conditions to activate the expression of *sigF* and so initiate sporulation. To validate that this control structure has been correctly modelled we formulate a range of tests on our Petri net model and compare the results with those presented in de Jong *et al*. (2003, 2004). We begin by running a simulation test in which the model's initial state is set to represent vegative growth but Signal is activated to represent the presence of nutritional stress. The sequence of states resulting from this simulation is presented in Table 3, where the first column of the table represents the model's initial state and each subsequent column the next state observed after a synchronized update. The simulation shows that SpoOA∼P accumulation is allowed to occur and SigF production is correctly activated indicating that the bacterium is able to sporulate. We then initialize our model so that Signal is inactive, representing the absence of nutritional stress, and simulate the model again. The results correctly show that sporulation does not occur, i.e. SpoOA∼P is not produced from the phosphorelay when Signal is inactive and so *sigF* is not expressed.

We can further validate the model using the model checking tools provided by PEP (Grahlmann, 1997; Khomenko, 2003) to confirm that basic properties of the genetic regulatory network are respected. For example, we know that in the absence of nutritional stress sporulation should never be initiated (de Jong *et al*., 2004). We can use model checking to show this holds in our model by proving that no reachable state exists with SigF = 1 starting from any initial state in which Signal = 0, SigF = 0 and SpoOA∼P = 0. Using a similar approach we can prove that sporulation can not occur in the absence of SigA (de Jong *et al*., 2004).

4.3.2 Property analysis Having gained some confidence in the correctness of our model from the validation process above we can now consider a more detailed analysis of the properties

Downloaded from https://academic.oup.com/bioinformatics/article/23/3/336/235741 by guest on 16 August 2022

Table 4. Simulation results starting from a state without nutritional stress

Entity	States			
SigF	0	0	0	0
KinA	0	0	1	0
SpoOA	1	1	1	1
SpoOA∼P	0	0	0	0
AbrB	0	1	0	1
SpoOE	0	1	0	1
SigH	0	1	0	1
Hpr	1	0	1	0
SinR	0	0	0	0
SinI	0	0	0	0
Signal	0	0	0	0

of the model. The idea is to begin by performing a range of simulations on the model, starting from biologically plausable initial states, and to then examine the results using standard techniques (Gershenson, 2002) to gain insight into the model's behaviour. As an example, consider the simulation shown in Table 4 which indicates there is a strong attractor cycle of period two (Wuensche, 2003) (last two columns in table) which correctly describes the physiological conditions for the exponential growth phase (de Jong *et al*., 2004).

Next we consider using our model to investigate experimental hypotheses which we formulate by examining the available experimental literature and using the insights gained from the simulations that have been performed. For example, in our case study the available literature indicates an important relationship between SigF and SpoOA∼P; the phosphorylated protein SpoOA∼P is reported to activate the production of SigF but also repress its own production (de Jong *et al*., 2004). Thus, it seems likely that the two entities should never be active at the same time, a property known as *mutual exclusion* (Murata, 1989). We verify this property by using model checking to show that no reachable state exists in our model in which SigF and SpoOA∼P are both active, specified by the following constraints:

$$(\mathrm{SigF} + \mathrm{SpoOA}\sim\mathrm{P}) = 2, \quad \mathrm{SinI\_Done} = 1$$

Note the condition SinI_Done = 1 is used here to ensure we only consider states reached after a complete pass of the two-phase-commit protocol, where SinI is the last entity to be updated.

The above model checking technique provides a template which can be used to test other types of relationships between entities in a model. For example, by inspecting the simulation results above (see Tables 3 and 4) it appears that SpoOE and AbrB always have the same state. We can attempt to verify this hypothesis by using model checking to see if a contradictory state can be reached, which we specify using the following constraints:

$$(\mathrm{SpoOE} + \mathrm{AbrB}) = 1, \quad \mathrm{SinI\_Done} = 1$$

Model checking shows that such a contradictory state is in fact reachable, disproving our hypothesis. The tool returns a transition firing sequence which leads to a contradictory state in which SpoOE and AbrB have different states as a counter example and this can be automatically simulated in PEP giving important insight into how this behaviour occurs.

Table 5. Results of mutant analysis

Entity	Knock out	Overexpressed
KinA	No sporulation	Normal
SpoOA	No sporulation	Normal
AbrB	No sporulation	No sporulation
SpoOE	Normal	No sporulation
SigH	No sporulation	Normal
SinR	Normal	No sporulation

4.3.3 Mutant analysis We complete the analysis by investigating the affect of 'fixing' a gene in the model to either be permanently active or inactive. This corresponds to the experimental approach of creating mutants in which genes are knocked out or overexpressed and is a useful analysis technique which can provide important insight into a genetic regulatory network. The idea is to construct a new model for each entity to be fixed where the input data used by the support tool has had the truth-table for the entity in question removed. As such the chosen entity will be treated as an input entity, as SigA and Signal are, and so its state will remain fixed once initialized. We have applied this technique to a range of entities in the sporulation model and then investigated how this affects the sporulation process using simulations and model checking. The observed results are summarised in Table 5 and correspond well with the experimental results available in the literature (see de Jong *et al*., 2004). Interestingly, we see that *abrB* is the only gene which has an affect when either knocked out or overexpressed. This fits with our understanding of *abrB* being a transition state regulator gene that lies at the center of three competing feedback loops (de Jong *et al*., 2004).

5 DISCUSSION

In this paper we have developed a new approach to qualitatively modelling genetic regulatory networks based on using the well–developed formal framework of Petri nets. The results we have presented significantly extend existing work on using Boolean models to analyse genetic regulatory networks (e.g. Chaouiya *et al*., 2004). In particular, the key contributions of this paper are as follows: (1) A new compact approach to qualitatively modelling genetic regulatory networks based on using logic minimization and Petri nets; (2) Both synchronous and asynchronous semantics of Boolean networks (Gershenson, 2002) are catered for; (3) Approach able to model incomplete and inconsistent data; (4) Provision of tool support to automate model construction; and (5) A detailed case study investigating the application of Petri net tools to analyse a genetic regulatory network.

The case study we have presented provides important insight into the practical application of Petri net analysis techniques for validating and investigating models of complex regulatory systems. It highlighted the important role that model checking techniques can play in investigating a range of interesting model properties such as mutual exclusion. More work is needed to extend the set of biologically relevant properties that can be tested and the use of temporal logic as a property description language appears promising here (Chabrier-River *et al*., 2004).

Downloaded from https://academic.oup.com/bioinformatics/article/23/3/336/235741 by guest on 16 August 2022

An important area not considered in this paper is the automatic extraction of truth-table definitions for entities in a genetic network. We are working to address this by linking our techniques with an existing network inference tool *SARGE* (Shaw *et al*., 2004). One drawback of using a Boolean model is that the high–level of abstraction used means we can fail to capture important dynamic network behaviour. We aim to address this by extending our work to multi–valued networks (Mishchenko and Brayton, 2002), with a particular emphasis on ensuring our Petri net models remain tractable to analysis. Finally, we intend to integrate our qualitative modelling tools into related work on stochastic Petri net modelling (Shaw *et al*., 2006) and so provide much needed support in this area.

ACKNOWLEDGEMENTS

We are very grateful to M. Koutny and V. Khomenko for many useful discussions and to the anonymous referees for their helpful comments. We thank the EPSRC for supporting R. Banks and the BBSRC for support via the Centre for Integrated Systems Biology of Ageing and Nutrition (CISBAN). Finally we acknowledge the support of the Newcastle Systems Biology Resource Centre.

Conflict of Interest: none declared.

REFERENCES

Akutsu,T. *et al*. (1999) Identification of genetic networks from small number of gene expression patterns under the Boolean network model. *Proc. Pacific Symp. Biocomput*., **4**, 17–28.

Billington,J. *et al*. (2003) The Petri net markup language: concepts, technology, and tools. In: van der Aalst,W. and Best,E. (eds), *Application and Theory of Petri Nets 2003, Proceedings of the 24th International Conference on the Application and Theory of Petri Nets*, Lecture Notes in Computer Science **2679**, Springer-Verlag, pp. 483–505.

Bower,J.M. and Bolouri,H. (2001) *Computational Modelling of Genetic and Biochemical Networks*. MIT Press.

Brayton,R., Hachtel,G., McMullen,C. and Sangiovanni-Vincentelli,A. (1984) *Logic Minimisation Algorithms for VLSI Synthesis*. Kluwer Academic Publishers.

Breeding,K.J. (1992) *Digital Design Fundamentals*. Prentice Hall.

Chabrier-Rivier,N. *et al*. (2004) Modeling and querying biomolecular interaction networks. *Theor. Comput. Sci*., **325**, 25–44.

Chaouiya,C., Remy,E., Ruet,P. and Thieffry,D. (2004) Qualitative modelling of genetic networks: From logical regulatory graphs to standard Petri nets. In: Cortadella,J. and Reisig,W. (eds), *Proceedings of the International Conference on the Application and Theory of Petri Nets*, Lecture Notes in Computer Science **3099**, Springer-Verlag, pp. 137–156.

Comet,J.-P., Klaudel,H. and Liauzu,S. (2005) Modeling Multi-valued Genetic Regulatory Networks Using High-Level Petri Nets. In: Ciardo,G. and Darondeau,P. (eds), *Proceedings of the International Conference on the Application and Theory of Petri Nets*, Lecture Notes in Computer Science **3536**, Springer-Verlag, pp. 208–227.

de Jong,H. *et al*. (2003) Genetic Network Analyzer: qualitative simulation of genetic regulatory networks. *Bioinformatics*, **19**, 336–344.

de Jong,H. *et al*. (2004) Qualitative simulation of the initiation of sporulation in *Bacillus subtilis*. *Bull. Math. Biol*., **66**, 261–299.

Esparza,J. (1994) Model checking using net unfoldings. *Sci. Comput. Programm*., **23**, 151–195.

Gershenson,C. (2002) Classification of random boolean networks. In: Standish,R.K. *et al*. (ed.), *Proceedings of the 8th International Conference on Artificial Life,* MIT Press, pp. 1–8.

Goss,P.J.E. and Peccoud,J. (1998) Quantitative modelling of stochastic systems in molecular biology by using stochastic Petri nets. *Proc. Natl Acad. Sci. USA*, **95**, 6750–6755.

Grahlmann,B. (1997) The PEP tool. In: *Proceedings of 9th International Conference on Computer Aided Verification*, Lecture Notes in Computer Science **1254**, Springer-Verlag, pp. 440–443.

Grossman,P. (2002) *Discrete Mathematics for Computing*. Palgrave MacMillan.

Heiner,M., Koch,I. and Voss,K. (2001) Analysis and simulation of steady states in metabolic pathways with Petri nets. In: Jensen,K. (ed.), *Workshop on Practical Use of Coloured Petri Nets and the CPN Tools (CPN'01)*. Aarhus University, pp. 15–34.

Huang,S. (1999) Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery. *J. Mol. Med*., **77**, 469–480.

Kauffman,S.A. (1969) Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol*., **22**, 437–467.

Khomenko,V. (2003) *Model Checking Based on Prefixes of Petri Net Unfoldings*. Ph. D. Thesis, School of Computing Science. University of Newcastle upon Tyne.

Matsuno,H. *et al*. (2000) Hybrid Petri net representation of gene regulatory network. *Pac. Symp. Biocomput*., **5**, 338–349.

Mishchenko,A. and Brayton,R. (2002) Simplification of non-deterministic multi-valued networks. In: *IEEE/ACM International Conference on Computer–Aided Design,* pp. 557–562.

Murata,T. (1989) Petri nets: properties, analysis and applications. *Proc. IEEE*, **77**, 541–580.

Peleg,M. *et al*. (2005) Using Petri net tools to study properties and dynamics of biological systems. *J. Am. Med. Inform. Assoc*., **12**, 181–199.

Reddy,V.N. *et al*. (1996) Qualitative analysis of biochemical reaction systems. *Comput. Biol. Med*., **26**, 9–24.

Reisig,W. (1985) *Petri Nets, An Introduction*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag.

Reisig,W. and Rozenberg,G. (1998) Lectures on Petri nets I: basic models. Advances in Petri Nets, Lecture Notes in Computer Science **1491**, Springer-Verlag.

Shaw,O.J. *et al*. (2004) SARGE: a tool for creation of putative genetic networks. *Bioinformatics*, **20**, 3638–3640.

Shaw,O.J. *et al*. (2006) Automatic parameterisation of Stochastic Petri net models of biological networks. *Electron. Notes Theor. Comput. Sci*., **151**, 111–129.

Simão,E. *et al*. (2005) Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in *E.Coli*. *Bioinformatics*, **21**, 190–196.

Spellman,P.T. *et al*. (1998) Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell*, **9**, 3273–3297.

Stephenson,K. and Lewis,R.J. (2005) Molecular insights into the initiation of sporulation in Gram-positive bacteria: new technologies for an old phenomenon. *FEMS Microbiol. Rev*., **29**, 281–301.

Stragier,P. and Losick,R. (1996) Molecular genetics of sporulation in *Bacillus subtilis*. *Ann. Revi. Genet*., **30**, 297–341.

Szallasi,Z. and Liang,S. (1998) Modeling the normal and neoplastic cell cycle with 'realistic Boolean genetic networks': their application for understanding carcinogenesis and assessing therapeutic strategies. *Pac. Symp. Biocomput*., **3**, 66–76.

Tsavachidou,D. and Liebman,M.N. (2002) Modeling and simulation of pathways in menopause. *J. Am. Med. Inform. Assoc*., **9**, 461–471.

Wuensche,A. (2003) Basins of attraction in network dynamics: a conceptual framework for biomolecular networks. In: Schlosser,G. and Wagner,G.P. (eds), *Modularity in Development and Evolution*. Chicago University Press, pp. 288–311.

Uetz,P. *et al*. (2000) A comprehensive analysis of protein–protein interactions in *Saccharomyces cerevisiae*. *Nature*, **403**, 623–627.

Downloaded from https://academic.oup.com/bioinformatics/article/23/3/336/235741 by guest on 16 August 2022