Modelling and Analysing Genetic Networks: From Boolean Networks to Petri Nets

L. J. Steggles, Richard Banks, and Anil Wipat

School of Computing Science, University of Newcastle, Newcastle upon Tyne, U. K.
{L.J.Steggles, Richard.Banks, Anil.Wipat}@ncl.ac.uk

Abstract. In order to understand complex genetic regulatory networks researchers require automated formal modelling techniques that provide appropriate analysis tools. In this paper we propose a new qualitative model for genetic regulatory networks based on Petri nets and detail a process for automatically constructing these models using logic minimization. We take as our starting point the Boolean network approach in which regulatory entities are viewed abstractly as binary switches. The idea is to extract terms representing a Boolean network using logic minimization and to then directly translate these terms into appropriate Petri net control structures. The resulting compact Petri net model addresses a number of shortcomings associated with Boolean networks and is particularly suited to analysis using the wide range of Petri net tools. We demonstrate our approach by presenting a detailed case study in which the genetic regulatory network underlying the nutritional stress response in *Escherichia coli* is modelled and analysed.

1 Introduction

The development and function of cellular systems is regulated by complex networks of interacting genes, proteins and metabolites known as *genetic regulatory networks* [3]. With the advent of improved post–genomic technology the data is now available to allow researchers to study genetic regulatory networks at a holistic level [26]. However, interpreting and analysing this data is still problematic and further work is needed to develop automated formal techniques that provide appropriate tools for modelling and analysing genetic regulatory networks.

In this paper, we present a new technique for qualitatively modelling and analysing genetic regulatory networks. We take as our starting point *Boolean networks* [1, 3], 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 regulatory networks [14, 27], 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* [21, 18], a well developed formal framework for modelling and analysing complex concurrent systems [22, 28]. A range of initial investigations

into using Petri nets to model biological systems have appeared in the literature to date, including: Place/Transition nets [19, 5, 25, 12]; stochastic nets [9, 24]; high–level nets [11, 6]; and hybrid nets [17]. The results we present significantly extend the related ideas presented in [5], both semantically and in the provision of automated tool support for model construction and analysis.

The Petri net model we propose is based on using an intuitive Petri net structure to represent the Boolean relationships between regulatory entities. We start by defining each entities individual behaviour as a *truth table* [10] from which we extract Boolean terms by applying *logic minimization techniques* [10, 4]. These Boolean terms compactly represent the fundamental relationships between regulatory entities and we directly translate them into appropriate Petri net control structures. The result is a compact Petri net model that completely captures the original Boolean behaviour of a genetic regulatory network. Both the *synchronous* and *asynchronous* semantic interpretation of Boolean networks [8] 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. To support the modelling process a prototype tool has been developed which is able to automatically construct Petri net models of genetic networks from their truth table definitions. The resulting models can then be analysed using the wide range of available Petri net techniques and tools [22, 7, 28].

We illustrate our approach by presenting a detailed case study in which the genetic regulatory network for the carbon starvation stress response in the bacterium *E. coli* [20] is modelled and analysed. Using the detailed data provided in [20] we define the Boolean behaviour of the key regulatory entities involved using truth tables. We then apply our prototype tool to automatically construct a qualitative Petri net model capturing the behaviour of the given genetic regulatory network. This Petri net is then validated and analysed using PEP [29] and in particular, we illustrate the application of model checking techniques [7, 15] for detailed model analysis.

This paper is organised as follows. In Section 2 we give a brief introduction to Boolean networks and 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 the carbon starvation stress response in *E. coli*. Finally, in Section 5, we present some concluding remarks on our work.

2 Background

In this section we give a brief overview of the modelling formalisms discussed in this paper: *Boolean networks* [1, 3] and *Petri nets* [21, 18]. In the sequel we assume the reader is familiar with the basic Boolean operators *not*, *or* and *and* (for example, see [10]).

2.1 Boolean Networks

In a Boolean network [1, 3] the state of each regulatory entity g_i 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 gene 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 [4]. 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 1.(a) [1] which contains three entities g_1, g_2 and g_3, where the next state g_i' of each entity is defined by the following Boolean functions:

$$g_1' = g_2, \quad g_2' = g_1\, g_3, \quad g_3' = \overline{g_1}$$

where the notation $\overline{x}$, $x+y$ and $x\ y$ is used to represent the Boolean operators *not*, *or* and *and* [10] respectively. The dynamic behaviour of a Boolean network can be semantically interpreted in two distinct ways [8]: *asynchronously* where genes update their state independently; and *synchronously* where all genes update their state together. We focus on the synchronous semantics in this paper which appears to be widely used in the literature [3, 8]. The synchronous behaviour for our example Boolean network is shown as a *truth table* in Figure 1.(b) and a *state transition graph* [4] in Figure 1.(c).

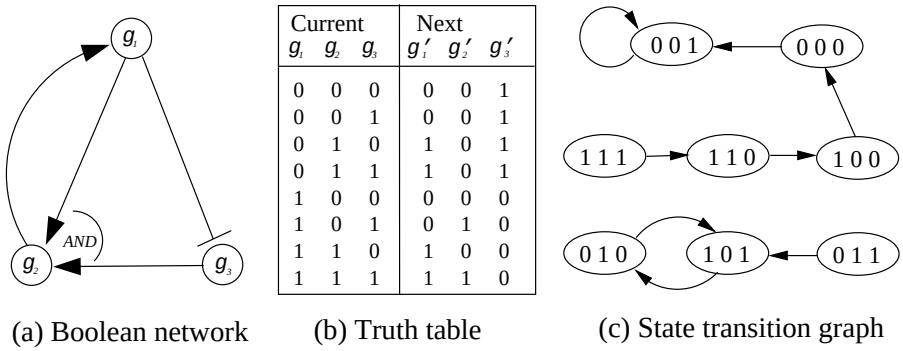

Fig. 1. An example of a Boolean network for three entities g_1, g_2 and g_3.

Boolean networks have proved successful in modelling real world regulatory networks [14, 27]. 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.

2.2 Petri Nets

The theory of Petri nets [21, 18] 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 2. The places, transitions and

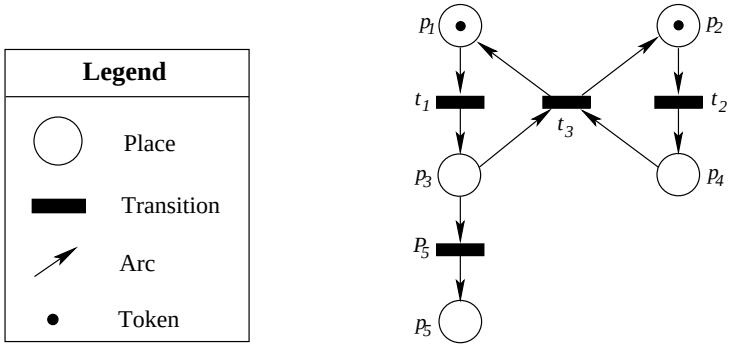

Fig. 2. A simple example of a Petri net.

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 [21]. 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 2 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. Often, more than one transition is enabled to fire at any one time (as in the example above). 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 $m1$ which results in the marking $m2$. 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 has a restricted state space which is well–suited to automatic analysis [22].

An important advantage of Petri nets is that they are supported by a wide range of techniques and tools for simulation and analysis [22, 28]. For example, Petri nets can be automatically checked for boundedness and the presence of deadlocks (markings in which no transitions are enabled to fire) [28]. A Petri net can also be analysed by constructing its *reachability graph* [18] which captures the possible firing sequences that can occur from a given initial marking. A range of techniques based on *model checking* [7, 15] have been developed for analysing reachability properties of a Petri net and these provide a means of coping with the potentially large state space of a Petri net model.

3 Modelling Genetic Networks using Petri Nets

In this section we present a new qualitative model for gene regulatory networks based on *Petri nets* [18] and detail a process for automatically constructing these models using *logic minimization* [4].

3.1 Deriving Regulatory Relationships using Logic Minimization

Given a set of truth tables defining the Boolean behaviour of all the entities in 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 [4, 10] 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, consider the truth table given in Figure 1.(b) for a simple Boolean network (see Section 2.1). Then by considering the truth table for g_1 we can see that states 010, 011, 110, and 111 result in g_1 being 1 in its next state (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* [10], using the *and* Boolean operator, where the variable g_i represents 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* [10] 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. In our example, we derive the following terms defining the behaviour of g_1, g_2 and g_3:

$$g_1' = \overline{g_1}\, g_2\, \overline{g_3} \;+\; \overline{g_1}\, g_2\, g_3 \;+\; g_1\, g_2\, \overline{g_3} \;+\; g_1\, g_2\, g_3,$$
$$g_2' = g_1\, \overline{g_2}\, g_3 \;+\; g_1\, g_2\, g_3,$$
$$g_3' = \overline{g_1}\, \overline{g_2}\, \overline{g_3} \;+\; \overline{g_1}\, \overline{g_2}\, g_3 \;+\; \overline{g_1}\, g_2\, \overline{g_3} \;+\; \overline{g_1}\, g_2\, g_3.$$

The Boolean terms derived above are often unnecessarily complex and can normally be simplified using *logic minimization* [4, 10]. 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 that differ by only one variable g_3. This term can be simplified by merging the two minterms to produce a simpler term $\overline{g_1}\, g_2$ which is logically equivalent [4, 10]. For brevity we omit the full details of Boolean logic minimization here (we refer the interested reader to [4]) and instead illustrate the idea behind the algorithm using our running example:

$$\overline{g_1}\, g_2\, \overline{g_3} \;+\; \overline{g_1}\, g_2\, g_3 \;+\; g_1\, g_2\, \overline{g_3} \;+\; g_1\, g_2\, g_3 \;\Longrightarrow\; \overline{g_1}\, g_2 \;+\; g_1\, g_2 \;\Longrightarrow\; g_2,$$
$$g_1\, \overline{g_2}\, g_3 \;+\; g_1\, g_2\, g_3 \;\Longrightarrow\; g_1\, g_3,$$
$$\overline{g_1}\, \overline{g_2}\, \overline{g_3} \;+\; \overline{g_1}\, \overline{g_2}\, g_3 \;+\; \overline{g_1}\, g_2\, \overline{g_3} \;+\; \overline{g_1}\, g_2\, g_3 \;\Longrightarrow\; \overline{g_1}\, \overline{g_2} \;+\; \overline{g_1}\, g_2 \;\Longrightarrow\; \overline{g_1}.$$

Note that the final minimized Boolean terms presented above correctly correspond to the Boolean network definitions given in Section 2.1.

3.2 Modelling Boolean Networks using Petri Nets

While the Boolean terms derived in Section 3.1 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 directly into appropriate Petri net control structures. The resulting Petri net model can then be simulated and analysed using the wide range of available tool support [28].

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 [21, 5]) 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 [5] for a related approach). We focus on modelling the synchronous behaviour of a Boolean network [8] 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 according to the recorded decision.

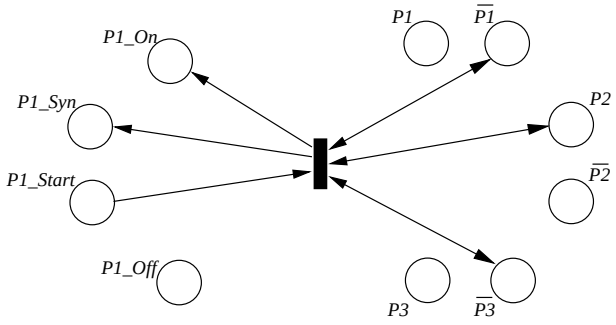

Fig. 3. A transition for gene g_1 modelling the minterm $\overline{g_1}\, g_2\, \overline{g_3}$.

Let us consider how we construct the appropriate Petri net structure to model the decision process for an entity g_i in the first phase of the protocol. We begin by considering under what conditions the entity will be active (i.e. in state 1) and use the process detailed in Section 3.1 to derive a minimized Boolean term which compactly captures these conditions. We model this minimized Boolean term in our Petri net by adding a separate transition to represent each minterm it contains. The idea is that each transition will fire, placing a token on Pi_On, precisely when the corresponding minterm is true. As an example, consider the Boolean term

$$\overline{g_1}\, g_2\, \overline{g_3} \;+\; \overline{g_1}\, g_2\, g_3 \;+\; g_1\, g_2\, \overline{g_3} \;+\; g_1\, g_2\, g_3$$

derived for gene g_1 in our running example (see Section 3.1). Then the first minterm $\overline{g_1}\, g_2\, \overline{g_3}$ tells us that gene g_1 should be expressed, $g_1 = 1$, in the next state when genes $g_1 = 0$, $g_2 = 1$, and $g_3 = 0$ in the current state. We model this minterm using the transition depicted in Figure 3. This transition fires when places $\overline{P1}$, $P2$, and $\overline{P3}$ contain a token (i.e. when $g_1 = 0$, $g_2 = 1$, and $g_3 = 0$) and results in a token being placed on $P1\_On$ (indicating g_1 is expressed in the next state). Note the use of *read arcs* [18] here, i.e. bidirectional arcs which do not consume tokens but just check they are present. This ensures the tokens on places $\overline{P1}$, $P2$, and $\overline{P3}$ are not removed at this stage (doing so would corrupt the current state of genes g_1, g_2 and g_3). The start place $P1\_Start$ is used as a control input to the transition to ensure only one decision is made for gene g_1 during a single protocol step (update transitions can only fire if a token is present on $P1\_Start$). The place $P1\_Syn$ is used to indicate when an update decision has been made for gene g_1, information needed by the protocol to determine when

the first phase is complete. This process is then repeated to add transitions to model the remaining three minterms in the Boolean term for g_1.

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 in Section 3.1 again amended to derive a Boolean term which compactly captures the conditions under which the entity becomes inactive. We then repeat the procedure detailed above for modelling a minterm as a transition except this time we mark place Pi_Off to record the decision for the next state instead of Pi_On. Note the resulting Petri net structure will contain at most $n2^k$ transitions where n is the number of entities and k is the maximum neighbourhood size. Since k is usually small in practice [8] the size of the model is normally linear with respect to n.

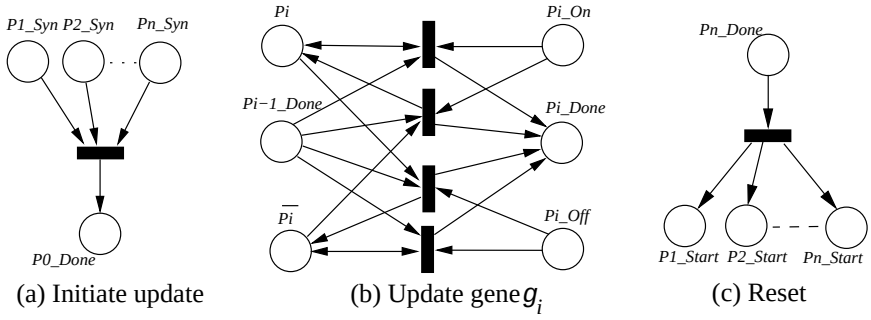

Fig. 4. Petri net fragments for controlling synchronous updates.

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 4.(a) 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 4.(b) for an arbitrary gene 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 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 4.(c) initiates a reset step which re-marks the start places, allowing the whole synchronisation protocol to begin again.

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 regulatory network. However, in practice it is rarely the case that a regulatory

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 about states. The result is that the behaviour of some entities under certain conditions may be unknown. Such incomplete and/or inconsistent information is problematic for the standard Boolean network model which is unable to represent the possibility of more than one next state. However, Petri nets are a non-deterministic modelling language [21] and so are able to represent unknown behaviour by incorporating all possible next state transitions. The idea is to simply allow the states with unknown behaviour to be used when deriving both the active and inactive Boolean formulas for an entity. The resulting non-deterministic choices within the Petri net model can then be meaningfully taken into account when analysing its behaviour.

The Petri net modelling approach presented above, while theoretically well–founded, is not practical by hand for all but the smallest of models. To support our modelling approach we have developed a prototype tool to automate the model construction process detailed in Sections 3.1 and 3.2. The tool takes as input a series of truth tables describing the behaviour of the entities in a Boolean network. These input tables are allowed to contain inconsistent and incomplete data as discussed above. From these tables the tool is able to automatically construct a Petri net model which is based on either the synchronous or asynchronous Boolean network semantics [8]. This prototype tool is freely available for academic use and can be obtained from the project's website[1].

4 Case Study: Nutritional Stress Response in *E. coli*

In this section we present a detailed case study to demonstrate the modelling techniques we have introduced and the practical application of Petri net analysis techniques. We consider the bacterium *E. coli* which under normal environmental conditions, when nutrients are freely available, is able to grow rapidly entering an *exponential phase* of growth [13]. However, as important nutrients become depleted and scarce the bacteria experiences nutritional stress and responds by slowing down growth, eventually resulting in a *stationary phase* of growth. In this case study we model a simplified version of the genetic regulatory network responsible for the carbon starvation nutritional stress response in *E. coli* based on the comprehensive data collated in [20]. We validate and analyse the resulting Petri net model using *PEP* [29], a leading Petri net support tool.

4.1 Constructing the Petri Net Model

The genetic regulatory network underlying the stress response in *E. coli* to carbon starvation is shown abstractly in Figure 5 (adapted from [20]). The network has a single input signal which indicates the presence or absence of carbon starvation and uses the level of stable RNA (ribosomal RNA and transfer RNA) as

[1] http://bioinf.ncl.ac.uk/gnapn

indicative of the current phase of *E. coli*, i.e. during the exponential phase the level of stable RNA is high to support rapid growth, while under the stationary phase the level drops, since only a maintenance metabolism is required [20]. The carbon starvation signal is transduced by the activation of adenylate cyclase (Cya), an enzyme which results in the production of the metabolite cAMP. This metabolite immediately binds with and activates the global regulator protein CRP, and the resulting cAMP.CRP complex is responsible for controlling the expression of key global regulators including Fis and CRP itself. The global regulatory protein Fis is central to the stress response and is responsible for promoting the expression of stable RNA from the *rrn operon* [13, 20]. Thus, during the exponential phase high levels of Fis are normally observed and the mutual repression that occurs between Fis and cAMP.CRP is thought to play a key role in the regulatory network [20]. The expression of *fis* is also promoted by high levels of *negative supercoiling* being present in the DNA. The level of DNA supercoiling is tightly regulated by two topoisomerases [13, 20]: GyrAB (composed of the products of genes *gyrA* and *gyrB*) which promotes supercoiling; and TopA which removes supercoils. An increase in DNA supercoiling results in increased expression of TopA and thus prevents excessive supercoiling. A decrease in supercoiling results in increased expression of *gyrA* and *gyrB*, and the resulting high level of GyrAB acts to increase supercoiling.

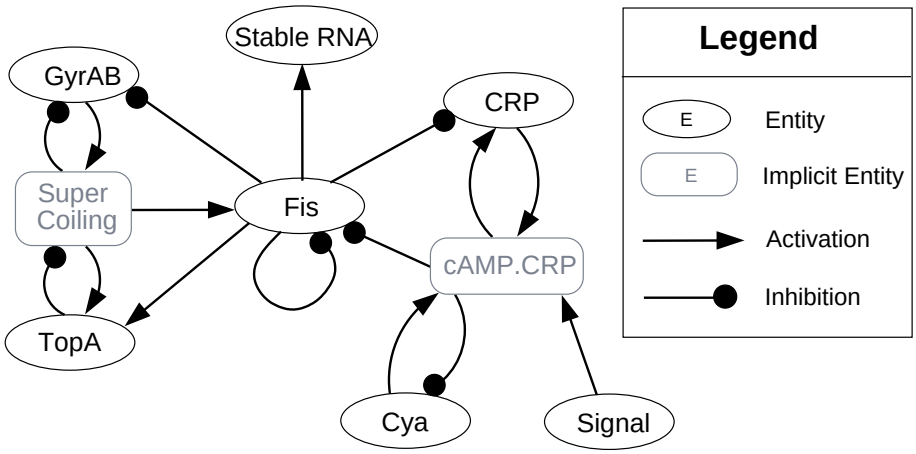

Fig. 5. Genetic network for carbon starvation stress response in *E. coli*.

Using the data provided in [20] we are able to derive truth tables defining the Boolean behaviour of each regulatory entity in the nutritional stress response network for carbon starvation. As an example, consider the truth table defining the behaviour of Cya shown in Figure 6 below. Note following the approach in [20], the level of cAMP.CRP and DNA supercoiling are not explicitly modelled as entities in our model.

CRP	Cya	Signal	Cya
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Fig. 6. Truth table defining the Boolean behaviour of Cya.

The next step is to apply logic minimization to the truth tables we have derived to extract Boolean expressions which compactly define the qualitative behaviour of each regulatory entity. This process is automated by our prototype tool and the result is the following set of Boolean equations:

$$\mathrm{Cya} = \overline{\mathrm{Signal}} \;+\; \overline{\mathrm{Cya}} \;+\; \overline{\mathrm{CRP}}, \quad \overline{\mathrm{Cya}} = \mathrm{Signal}\ \ \mathrm{Cya}\ \ \mathrm{CRP},$$

$$\mathrm{CRP} = \overline{\mathrm{Fis}}, \quad \overline{\mathrm{CRP}} = \mathrm{Fis},$$

$$\mathrm{GyrAB} = (\overline{\mathrm{GyrAB}}\ \ \overline{\mathrm{Fis}}) \;+\; (\mathrm{TopA}\ \ \overline{\mathrm{Fis}}),$$
$$\overline{\mathrm{GyrAB}} = (\mathrm{GyrAB}\ \ \overline{\mathrm{TopA}}) \;+\; \mathrm{Fis},$$

$$\mathrm{TopA} = \mathrm{GyrAB}\ \ \overline{\mathrm{TopA}}\ \ \mathrm{Fis}, \quad \overline{\mathrm{TopA}} = \overline{\mathrm{GyrAB}} \;+\; \mathrm{TopA} \;+\; \overline{\mathrm{Fis}},$$

$$\mathrm{Fis} = (\overline{\mathrm{Fis}}\ \ \overline{\mathrm{Signal}}\ \ \mathrm{GyrAB}\ \ \overline{\mathrm{TopA}}) \;+\; (\overline{\mathrm{Fis}}\ \ \overline{\mathrm{Cya}}\ \ \mathrm{GyrAB}\ \ \overline{\mathrm{TopA}})$$
$$+\; (\overline{\mathrm{Fis}}\ \ \overline{\mathrm{CRP}}\ \ \mathrm{GyrAB}\ \ \overline{\mathrm{TopA}}),$$
$$\overline{\mathrm{Fis}} = (\mathrm{CRP}\ \ \mathrm{Cya}\ \ \mathrm{Signal}) \;+\; \mathrm{Fis} \;+\; \overline{\mathrm{GyrAB}} \;+\; \mathrm{TopA},$$

$$\mathrm{SRNA} = \mathrm{Fis}, \quad \overline{\mathrm{SRNA}} = \overline{\mathrm{Fis}}.$$

The above equations can then be used to construct a Petri net model of the nutritional stress response regulatory network for carbon starvation by applying the approach detailed in Section 3.2. The result is a safe Petri net model that contains 45 places and 49 transitions (based on the synchronous update semantics). The above process can be automated using our prototype tool and the resulting Petri net can then be exported to a wide range of Petri net tools [28].

4.2 Analysing the Petri Net Model

We now consider analysing the Petri net model which results above using the PEP tool [29] and in particular, make use of model checking techniques [7, 15].

Our aim is to illustrate the range of analysis possible using available tools, from simple validation tests to more in–depth gene 'knockout' analysis.

We begin our analysis by performing a series of simple validation tests to check the model is able to correctly switch between the exponential and stationary phases of growth. The idea is to initialise the Petri net to a given state and then simulate it, observing the states that occur after each application of the two phase commit protocol. The results of these simulations can then be compared with the expected behaviour [13, 20, 2] to validate the model. As an example, we consider validating that the model correctly switches from the exponential to the stationary phase of growth. We initialise the Petri net to a state representing the exponential phase but activate Signal to represent the presence of carbon starvation. The resulting simulation run is presented in Figure 7, where the first row represents the models initial state and each subsequent row the next state observed. It shows that the model correctly switches to the stationary phase by entering an attractor cycle of period two (see last three rows in table) in which stable RNA is not present in significant levels (i.e. SRNA remains inactive).

Signal	CRP	Cya	GyrAB	TopA	Fis	SRNA
1	0	1	1	0	1	1
1	0	1	0	1	0	1
1	1	1	1	0	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	0

Fig. 7. Simulating the switch from exponential to stationary phase.

To investigate the behaviour of the model in more detail we make use of the extended reachability analysis provided by the model checking tools of PEP [7, 15]. For example, it appears from the literature that the entities GyrAB and TopA should be mutually exclusive, i.e. whenever GyrAB is significantly expressed then TopA shouldn't be and vice a versa. We can verify this in our model by formulating the following constraint on places:

$$\text{GyrAB} + \text{TopA} \; > \; 1, \; \text{GyrAB\_Done} \; = \; 1$$

which characterises a state in which the mutual exclusion property does not hold (where the condition GyrAB_Done = 1 is used to ensure we only consider states reached after a complete pass of the two phase commit protocol). The model checking tool is able to confirm that no state satisfying this constraint is reachable from any reasonable initial state and this proves that GyrAB and TopA must be mutually exclusive. We can attempt to prove a similar mutual exclusion property for CRP and Fis using the same approach. However, this time the model checking tool confirms that it is able to reach a state satisfying the

constraint, proving that CRP and Fis are not mutually exclusive in our model. In fact, the tool returns a witness firing sequence which leads to such a state to validate the result and we are able to automatically simulate this to gain important insight into how this behaviour occurs.

We can extend our analysis further by experimenting with the underlying structure of the Petri net model, adding or removing regulatory relationships to test possible experimental hypotheses. To illustrate this we can consider investigating the effect of fixing the level of the global regulator CRP which is the target of the carbon starvation signal–transduction pathway [20]. We do this by simply omitting the truth table for CRP from the construction process, resulting in CRP being treated as an input entity (i.e. like the entity Signal) whose state becomes fixed once initialised. We start by 'knocking out' *crp* so that it cannot be expressed and then simulate the amended model to investigate the impact of this change. As expected the results show that the transition from exponential to stationary phase is blocked; the lack of CRP prevents the formation of cAMP.CRP which is needed to initiate the phase transition. Next we fix *crp* to be permanently expressed and again simulate the model. Interestingly the results show that the behaviour of the network is largely unaffected by this change; both the transition from exponential to stationary phase and vice a versa are able to occur as normal.

5 Conclusions

The standard approach of using Boolean networks [1, 3] to model genetic regulatory networks has a number of shortcomings: Boolean networks lack effective analysis tools; and have problems coping with incomplete or inconsistent data. In this paper we addressed the shortcomings of Boolean networks by presenting a new approach for qualitatively modelling genetic regulatory networks based on Petri nets [18]. The idea was to use logic minimization [4] to extract Boolean terms representing the genetic network's behaviour and to then directly translate these into Petri net control structures. The result is a compact Petri net model that correctly captures the dynamic behaviour of the original regulatory network and which is amenable to detailed analysis via existing Petri net tools [28].

We illustrated our approach by modelling and analysing the genetic regulatory network underlying the carbon starvation stress response in *E. coli* [20, 2]. This case study demonstrated how the PEP tool [29] can be used to validate and analyse our Petri net models. In particular, we considered using simulation tests to validate the correctness of our model and model checking tools [29, 15] to investigate the detailed behaviour of the genetic regulatory network.

The results we have presented significantly extend existing work on using Boolean models to analyse genetic regulatory networks (e. g. [5]). In particular, we see the key contributions of this paper as follows: i) A new compact approach to qualitatively modelling genetic regulatory networks based on using logic minimization and Petri nets; ii) Both synchronous and asynchronous semantics of

Boolean networks [8] are catered for; iii) Provision of tool support to automate model construction; iv) A detailed case study exploring the application of existing Petri net tools to analyse a Boolean model of a genetic regulatory network.

One drawback of Boolean models is that the high level of abstraction used means behaviour crucial to the operation of a regulatory network may be lost. In future work we intend to address this problem by extending our modelling approach to multi–valued network models [16]. We intend to incorporate our qualitative modelling tools into related work on Stochastic Petri net modelling [23, 24] and so provide much needed support in this important area.

Acknowledgments. We are very grateful to O. J. Shaw, M. Koutny and V. Khomenko for many useful discussions concerning this work. We would also like to thank the EPSRC for supporting R. Banks and the BBSRC for supporting this work via the Centre for Integrated Systems Biology of Ageing and Nutrition (CISBAN). Finally we acknowledge the support of the Newcastle Systems Biology Resource Centre.

References

1. T. Akutsu, S. Miyano and S. Kuhara. Identification of genetic networks from small number of gene expression patterns under the Boolean network model. *Proceedings of Pacific Symp. on Biocomputing*, 4:17-28, 1999.
2. G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page and D. Schneider. Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in *Escherichia coli*. *Bioinformatics*, 21:i19–i28, 2005.
3. J. M. Bower and H. Bolouri. *Computational Modelling of Genetic and Biochemical Networks*. MIT Press, 2001.
4. K. J. Breeding. *Digital Design Fundamentals*. Prentice Hall, 1992.
5. C. Chaouiya, E. Remy, P. Ruet, and D. Thieffry. Qualitative modelling of genetic networks: From logical regulatory graphs to standard Petri nets. In: J. Cortadella and W. Reisig (Eds), *Proc. of the Int. Conf. on the Application and Theory of Petri Nets*, Lecture Notes in Computer Science 3099, pages 137-156, Springer–Verlag, 2004.
6. J.-P. Comet, H. Klaudel and S. Liauzu. Modeling Multi-valued Genetic Regulatory Networks Using High-Level Petri Nets. In: G. Ciardo and P. Darondeau (eds), *Proc. of the Int. Conf. on the Application and Theory of Petri Nets*, Lecture Notes in Computer Science 3536, pages 208-227, Springer–Verlag, 2005.
7. J. Esparza. Model checking using net unfoldings. *Science of Computer Programming*, 23(2-3):151-195, 1994.
8. C. Gershenson. Classification of random boolean networks. In: R. K. Standish et al (eds), *Proc. of the 8th Int. Conf. on Artificial Life*, p.1–8, MIT Press, 2002.
9. P. J. E. Goss and J. Peccoud. Quantitative modelling of stochastic systems in molecular biology by using stochastic Petri nets. *Proceedings of the National Academy of Sciences of the United States of America*, 95(12):6750-6755, 1998.
10. P. Grossman. *Discrete Mathematics for Computing*. Palgrave MacMillan, Second Edition, 2002.

11. M. Heiner, I. Koch, and K. Voss. Analysis and simulation of steady states in metabolic pathways with Petri nets. In K. Jensen (ed), *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (CPN'01)*, pages 15-34, Aarhus University, 2001.
12. M. Heiner, I. Koch, and J. Will. Model validation of biological pathways using Petri nets - demonstrated for apoptosis. *Biosystems*, 75(1-3):15-28, 2004.
13. R. Hengge-Aronis. The general stress response in *Escherichia coli*. In: G. Storz and R. Hengge-Aronis (eds), *Bacterial Stress Responses*, pages 161-178, ASM Press, 2000.
14. S. Huang. Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery. *Journal of Molecular Medicine*, Vol. 77:469-480, 1999.
15. V. Khomenko. *Model Checking Based on Prefixes of Petri Net Unfoldings*. Ph. D. Thesis, School of Computing Science, University of Newcastle upon Tyne, 2003.
16. B. Luque and F. J. Ballesteros. Random Walk Networks. *Physica A: Statistical Mechanics and its Applications*, 342(1-2):207-213, 2004.
17. H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid Petri net representation of gene regulatory network. *Pacific Symposium on Biocomputing*, 5:338-349, 2000.
18. T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
19. V. N. Reddy, M. N. Liebman, M. L. Mavrovouniotis. Qualitative analysis of biochemical reaction systems. *Computers in Biology and Medicine*, 26(1):9-24, 1996.
20. D. Ropers, H. de Jong, M. Page, D. Schneider, and J. Geiselmann. Qualitative Simulation of the Nutritional Stress Response in *Escherichia coli*. INRIA, Rapport de Reacherche no. 5412, December 2004.
21. W. Reisig. *Petri Nets, An Introduction*. EATCS Monographs on Theoretical Computer Science, W.Brauer *et al* (Eds.), Springer–Verlag, Berlin, 1985.
22. W. Reisig and G. Rozenberg. *Lectures on Petri Nets I: Basic Models*. Advances in Petri Nets, Lecture Notes in Computer Science 1491, Springer-Verlag, 1998.
23. O. J. Shaw, C. Harwood, A. Wipat, and L. J. Steggles. SARGE: A tool for creation of putative genetic networks. *Bioinformatics*, 20(18):3638-3640, 2004.
24. O. J. Shaw, L. J. Steggles, and A. Wipat. Automatic Parameterisation of Stochastic Petri Net Models of Biological Networks. *Electronic Notes in Theoretical Computing Science*, 151(3):111-129, June 2006.
25. E. Simão, E. Remy, D. Thieffry, C. Chaouiya. Qualitative Modelling of Regulated Metabolic Pathways: Application to the Tryptophan Biosynthesis in *E. Coli*. *Bioinformatics* 21:190-196, 2005.
26. P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273-3297, 1998.
27. Z. Szallasi and S. Liang. Modeling the Normal and Neoplastic Cell Cycle with "Realistic Boolean Genetic Networks": Their Application for Understanding Carcinogenesis and Assessing Therapeutic Strategies. *Pacific Symposium on Biocomputing* Vol. 3: 66–76, 1998.
28. *Petri nets World*, `http://www.informatik.uni-hamburg.de/TGI/PetriNets/`, 2006.
29. *PEP Home Page*, `http://parsys.informatik.uni-oldenburg.de/~pep/`, 2006.

This article was processed using the LaTeX macro package with LLNCS style