

# A High-Level Petri Net Framework for Genetic Regulatory Networks

Richard Banks and L. Jason Steggle

School of Computing Science, University of Newcastle, Newcastle upon Tyne, UK,  
<http://www.ncl.ac.uk>

## Summary

To understand the function of genetic regulatory networks in the development of cellular systems, we must not only realise the individual network entities, but also the manner by which they interact. *Multi-valued networks* are a promising qualitative approach for modelling such genetic regulatory networks, however, at present they have limited formal analysis techniques and tools. We present a flexible formal framework for modelling and analysing multi-valued genetic regulatory networks using *high-level Petri nets* and *logic minimization techniques*. We demonstrate our approach with a detailed case study in which part of the genetic regulatory network responsible for the carbon starvation stress response in *Escherichia coli* is modelled and analysed. We then compare and contrast this multi-valued model to a corresponding Boolean model and consider their formal relationship.

## 1 Introduction

To understand the function of genetic regulatory networks in the development of cellular systems, we must not only realise the individual network entities (i.e. genes, proteins and metabolites), but also the manner in which they interact. Given that many data resources are incomplete and inaccurate, the development and application of qualitative modelling techniques has emerged as an important topic of research [6]. A range of qualitative modelling techniques can be found in the literature, a few examples of which include: *Boolean networks* [30, 1, 27], *differential equations* [25] and *Petri nets* [23, 4, 29, 13, 18].

*Multi-valued networks* [26] are a promising qualitative approach for modelling genetic regulatory networks [31], providing a compromise between the simplicity of Boolean networks and the more detailed differential equational models. The idea is to extend the Boolean network approach by allowing the state of each regulatory entity to be represented by a range of discrete values. However, at present multi-valued networks appear to have limited formal techniques and tools to comprehensively analyse a genetic regulatory model. In addition, the formal relationship between multi-valued models and their Boolean counterparts is not well understood, for example, is often unclear when the Boolean approach is insufficient for a modelling task.

In this paper, we begin to address these concerns by extending and improving upon the Boolean approach of [29]. We develop a generalized approach for modelling and analysing multi-valued genetic regulatory networks using *high-level Petri nets* [2, 15]. High-level Petri nets are a well-developed formalism for modelling concurrent systems that are supported by a wide range of analysis techniques and tools [24]. We detail an approach for translating multi-valued models in to high-level Petri nets and in particular, make use of *logic minimization techniques* [19, 26] to ensure the Petri net models are as compact as possible. The result is a flexible formal framework for modelling and analysing multi-valued genetic networks which incorporates both

the *synchronous* and *asynchronous* network update semantics [9], and which is able to cope with the incomplete data that often occurs in practice.

We demonstrate our approach with a detailed case study in which part of the genetic regulatory network responsible for the carbon starvation stress response in *E. coli* [25, 14] is modelled and analysed. In particular, we aim to illustrate the type of analysis possible on our Petri net models, from simple simulation tests to more detailed mutant analysis based on model checking techniques [7, 17]. We compare and contrast our multi-valued case study model to a corresponding Boolean model [28] and identify some subtle discrepancies between the two. This leads to a number of interesting questions concerning the relationship between Boolean and multi-valued models, and we present some initial work on formally investigating this relationship.

The rest of this paper is structured as follows. In Section 2, we give a brief introduction to the theory of high-level Petri nets. In Section 3, we introduce multi-valued networks and multi-valued logic minimization. In Section 4, we describe a framework for modelling multi-valued genetic regulatory networks using high-level Petri nets, and then illustrate the approach with a case study in Section 5. In Section 6, we compare and contrast the multi-valued and Boolean network approaches, and present an initial investigation into the formal relationship between them. Finally, Section 7 presents some concluding remarks.

## 2 High-Level Petri Nets

*Petri nets* [21, 20] are a well-founded formalism for modelling and reasoning about concurrent, distributed systems that have been extensively used in computing science [22]. This section provides a simplified introduction to HLPNs appropriate for this paper; for a more detailed introduction we recommend [2, 15].

A *high-level Petri net* (HLPN) [8, 2, 15] is a directed, bipartite graph consisting of: *places*; *transitions*; and *arcs* connecting places and transitions. An example of a HLPN is presented in Figure 1. The state of a HLPN is represented by the *tokens* associated with the places of the net. Tokens represent data values and each place has a type which specifies the tokens it can hold. For example, in Figure 1 place  $p2$  has token type  $\{0..5\}$  restricting tokens for that place to numbers between 0 and 5, and we observe that  $p2$  currently contains two tokens, namely 2 and 3. Note that multiple copies of the same token are allowed on places, i.e. places contain multi-sets of tokens. The state of the whole HLPN is represented by a *marking* which maps places to the multi-set of tokens they contain. In Figure 1, the current marking  $M$  is defined by  $M(p1) = \{1\}$ ,  $M(p2) = \{2, 3\}$  and  $M(p3) = \{\}$ .

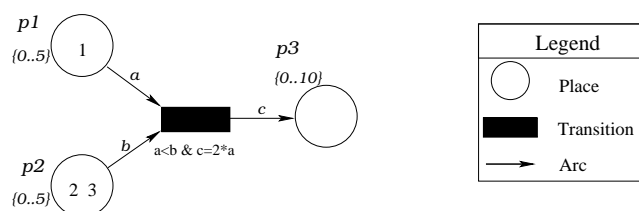


Figure 1: An example of a simple high-level Petri net.

The dynamic behaviour of a HLPN is modelled by transitions which are able to consume and produce new tokens. Each arc has an associated variable which *binds* to the token of the place it comes from or goes to. Each transition has a Boolean expression called a *guard* which

evaluates to either true or false for a given binding. A binding is said to *enable* a transition if: i) the token value assigned to each input variable resides on the associated input place; and ii) the transition's guard evaluates to true with the binding. An enabled transition may *fire* by removing tokens from each of its input places and adding a new token to each output place as specified by the enabling binding. Note if more than one transition is enabled then a transition is chosen non-deterministically to fire.

As an example, consider Figure 1 which contains a transition enabled by the binding  $\{a \mapsto 1, b \mapsto 3, c \mapsto 2\}$ . The transition can therefore fire resulting in token 1 and 3 being removed from places  $p_1$  and  $p_2$  respectively, and a new token 2 being added to place  $p_3$ . The net will now have a new marking  $M'$  defined by  $M'(p_1) = \{\}$ ,  $M'(p_2) = \{2\}$  and  $M'(p_3) = \{2\}$ .

A marking  $M_2$  is said to be *reachable* from a marking  $M_1$  if there exists a sequence of transitions that can be fired starting from  $M_1$  that result in  $M_2$ . The markings reachable in a HLPN can be analysed by constructing its *reachability graph* [20] which captures the possible firing sequences that can occur from a given initial marking. A range of techniques based on *model checking* [7, 17] have been developed for efficiently analysing reachability properties.

### 3 Multi-Valued Networks

A *multi-valued network* [26] consists of a set of nodes  $G = \{g_1, g_2, \dots, g_n\}$  representing *regulatory entities*. Each entity  $g_i$  has an associated state space  $S_{g_i} = \{0, 1, \dots, k_i\}$ , and we denote by  $\hat{g}_i \in S_{g_i}$  the current state of  $g_i$ . Furthermore, each entity  $g_i$  has a neighbourhood  $N(g_i) \subseteq G$  of entities that can affect its state. The dynamic behaviour of  $g_i$  is defined by a next state function which given the state of entities in  $N(g_i)$  defines the next state of  $g_i$ . Note the well-known *Boolean network* [16, 1, 30] approach is simply a special case of multi-valued networks in which  $S_{g_i} = \{0, 1\}$  for all entities  $g_i$ .

The next state function associated with each entity can be defined using a state transition table [26], which for each input state describes the next state the entity will enter. There are two possible state update semantics [9]: *synchronous* where all entities update in unison; and *asynchronous* where entities update independently. In this paper we concentrate on the synchronous semantics which is widely used in the biological community. As an example, consider Figure 2 which presents a simple multi-valued network consisting of three entities  $g_1, g_2$  and  $g_3$ , with state spaces  $S_{g_1} = \{0, 1, 2\}$ ,  $S_{g_2} = \{0, 1, 2\}$  and  $S_{g_3} = \{0, 1\}$ .

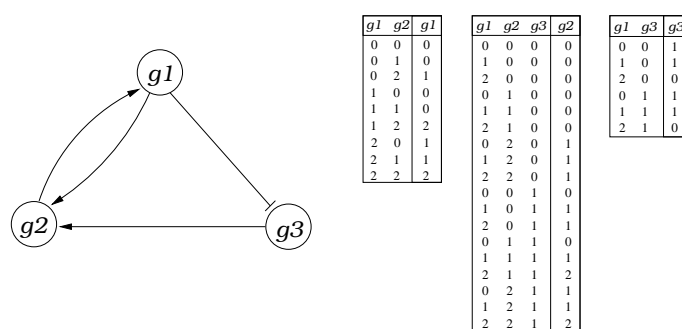


Figure 2: Multi-valued network with three entities  $g_1, g_2$  and  $g_3$ .

The state transition tables defining the next state function for each entity can be specified equationally [26, 19] by using Boolean terms called *literals* to formalise when an entity  $g_i$  is in one

of a set of states. Literals have the form  $g_i S$ , where  $S \subseteq S_{g_i}$ , and are defined to evaluate to true when  $\hat{g}_i \in S$  and to false otherwise. These literals can be combined into *product terms* using the Boolean operator *and* and these can then be used to represent input states in the state transition table. For example, from the table in Figure 2 we can see that  $g_1$  will have next state 0 when we are in state  $g_1 = 0, g_2 = 0$ , in other words, when product term  $g_1\{0\}g_2\{0\}$  is true. We can combine all the product terms representing the states that result in a particular next state using the *or* (denoted +) Boolean operator and the resulting term in *disjunctive normal form* [12] can then be used to equationally specify that next state. Continuing with our example, we derive the following equations which completely specify when  $g_1$ 's next state will be 0, 1 and 2:

$$\begin{aligned} g_1\{0\} &= g_1\{0\}g_2\{0\} + g_1\{0\}g_2\{1\} + g_1\{1\}g_2\{0\} + g_1\{1\}g_2\{1\}. \\ g_1\{1\} &= g_1\{0\}g_2\{2\} + g_1\{2\}g_2\{0\} + g_1\{2\}g_2\{1\}, \\ g_1\{2\} &= g_1\{1\}g_2\{2\} + g_1\{2\}g_2\{2\}. \end{aligned}$$

The equational descriptions above can normally be simplified using *multi-valued logic minimization* [26, 19], a process which syntactically simplifies logical terms while preserving their meaning. This process is important to our work as it facilitates a compact equational representation of the functional behaviour of multi-valued networks which can then be translated into a Petri net model. As an example, consider the first equation above which defines when the next state of entity  $g_1$  will be 0 in our running example. The first two terms can be combined together because they differ in only one literal  $g_2$ , as can the last two terms. To complete the simplification for when  $g_1$  goes to state 0 we combine the remaining two terms as they differ only by literal  $g_1$ . These steps are illustrated below:

$$\begin{aligned} g_1\{0\} &= g_1\{0\}g_2\{0, 1\} + g_1\{1\}g_2\{0\} + g_1\{1\}g_2\{1\} \\ g_1\{0\} &= g_1\{0\}g_2\{0, 1\} + g_1\{1\}g_2\{0, 1\}. \\ g_1\{0\} &= g_1\{0, 1\}g_2\{0, 1\} \end{aligned}$$

This systematic process can be repeated for when  $g_1$  goes to state 1 and 2, resulting in the following minimized equations:

$$\begin{aligned} g_1\{1\} &= g_1\{0\}g_2\{2\} + g_1\{2\}g_2\{0, 1\} \\ g_1\{2\} &= g_1\{1, 2\}g_2\{2\}. \end{aligned}$$

This logic minimization process can be automated using the *MVSIS* tool, which can be downloaded at <http://embedded.eecs.berkeley.edu/Respep/Research/mvsis>.

## 4 Modelling Multi-Valued Networks using Petri Nets

The compact equations derived through logic minimization completely describe the regulatory dynamics of a system, but are not amenable to analysis. In order to be able to understand and investigate the regulatory behaviour they specify, we propose a method for translating these equational descriptions into a HLPN model by building on existing work for Boolean networks (see [4, 28, 29]). Our systematic approach to HLPN model construction can be seen as complementary to [5], which focuses on using temporal logics to assist in the discovery of coherent HLPN models of partial multi-valued models.

The basic idea is to model each entity  $g_i$  in a multi-valued network as a place in the HLPN whose token type corresponds to the set of states  $S_{g_i}$  for that entity. A single transition with an appropriate guard is then used to capture the dynamic synchronous behaviour of the multi-valued network. Each place  $g_i$  communicates with this transition using two arcs: one going from the transition to the place labelled  $n(g_i)$ ; and one going from the place to the transition labelled  $c(g_i)$ . As an example, consider Figure 3 which depicts the topology of the HLPN for the multi-valued network shown in Figure 2.

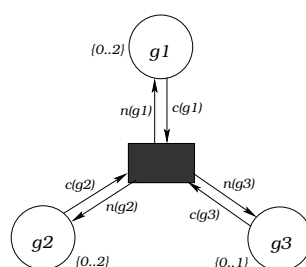


Figure 3: High-level Petri net for the multi-valued network shown in Figure 2.

The key step in modelling the dynamical behaviour of the multi-valued network is the correct construction of the guard for the single transition in the HLPN model. The idea is to translate the compact equational specification of the network derived using logic minimization into an appropriate transition guard. This approach takes each entity in turn, and translates its equations into a corresponding guard formula. The resulting formulas are then composed using the *and* (& in HLPN syntax [2]) Boolean operator. To illustrate this process we consider constructing the guard for the multi-valued network example in Figure 2. The formula will have the form:

$$enc(g_1) \& enc(g_2) \& enc(g_3)$$

where  $enc(g_i)$  represents the sub-formula for entity  $g_i$ . Taking  $g_1$  as an example,  $enc(g_1)$  will include three parts, one for each of its next states which are combined using the *or* Boolean operator (denoted |). Thus, it has the form:

$$enc(g_1\{0\}) \mid enc(g_1\{1\}) \mid enc(g_1\{2\})$$

where  $enc(g_i\{j\})$  represents the encoded behaviour of entity  $g_i$  at next state  $j$ . Each sub-formula  $enc(g_i\{j\})$  is then constructed by simply translating into HLPN syntax the right hand side of the corresponding simplified equation derived from the process in Section 3. For example, the sub-formula  $enc(g_1\{0\})$  is based on translating the right hand side of the equation for when  $g_1$  updates to state 0 (see Section 3), resulting in the following HLPN formula:

$$(n(g_1) = 0) \& ((c(g_1) = 0) \mid (c(g_1) = 1)) \& ((c(g_2) = 0) \mid (c(g_2) = 1)).$$

The above systematic construction results in a transition guard which describes the entire synchronous system behaviour. The resulting HLPN is then amenable to a range of analysis techniques [22], such as model checking [17]. A prototype tool to automate the Petri net construction from state transition tables is freely available for academic use [10]. This automation is important as it allows large models not amenable to manual construction to be automatically generated, allowing biologists to focus on model analysis.

Often, the data provided for a model is incomplete or inconsistent. Following the approach of [28], we simply make use of the non-deterministic semantics of Petri nets to cope with this by including all possible behaviours.

## 5 Case Study: Carbon Starvation Response in *E. coli*

This section provides a detailed case study in which a range of Petri net techniques are used to model and analyse the genetic regulatory network responsible for the carbon starvation stress response in *E. coli* [14].

### 5.1 Response to Carbon Starvation in *E. coli*

Under normal conditions with sufficient nutrient availability, the bacterium *E. coli* is able to develop rapidly entering an *exponential growth phase* [14]. However, under adverse conditions, when the nutrient availability is depleted, *E. coli* enters a *stationary phase* in which a substantial slow down in growth occurs to help the bacteria survive. The genetic network underlying this response for carbon starvation is shown abstractly in Figure 4 (adapted from [25]).

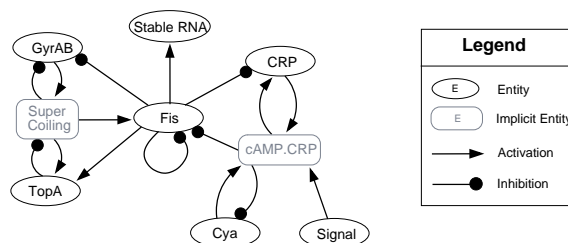


Figure 4: High-level regulatory network for the carbon starvation response network in *E. coli*.

The network has a single input signal indicating the presence of carbon starvation, which 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* [14, 25]. Thus, during the exponential growth 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 [25]. 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* [14, 25]: 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. For a more detailed introduction to the carbon starvation stress response in *E. coli* see [25, 14].

### 5.2 Constructing the HLPN Model

From the comprehensive data provided in [25], we are able to derive a set of state transition tables describing the multi-valued behaviour of each entity in the carbon starvation stress response network. Note following the approach of [25], we do not explicitly model the level of cAMP.CRP and DNA supercoiling as entities in our model. We then apply the techniques discussed in Section 3 to the state transition tables (using our tool support) to extract the simplified

equational specifications for the network. The simplified equations are then used to construct a HLPN model following the approach detailed in Section 4. As an example, consider the state

<i>Fis</i>	0	0	1	1	2	2	3	3	4	4	5	5
<i>RRN</i>	0	1	0	1	0	1	0	1	0	1	0	1
<i>RRN</i>	0	0	0	0	0	0	1	1	1	1	1	1

Figure 5: State transition table defining stable RNA behaviour.

transition table defining the level of stable RNA (denoted by *RRN*) with respect to *Fis* shown in Figure 5 which results in the simplified equations shown below:

$$\begin{aligned}
 RRN\{0\} &= Fis\{0, 1\} + Fis\{0, 2\} \\
 RRN\{1\} &= Fis\{3, 4, 5\}
 \end{aligned}$$

A complete listing of these equations can be found in the supplementary material. In addition, the resulting HLPN model is omitted for brevity but is available at [10].

### 5.3 Model Analysis

We now consider analysing our HLPN model using a range of Petri net techniques and tools. Our aim is to illustrate the type of analysis possible on our model, from simple validation tests to more detailed mutant analysis.

#### (i) Validation

The first step in the analysis process is to validate our model to ensure it is a reasonable representation of the genetic network under investigation. We do this by performing a range of simple tests on the model to ensure it satisfies the basic properties detailed in the literature (see [14, 25]). For example, we can check that our model correctly switches from exponential to stationary growth phases in the presence of carbon stress. To do this we performed a simulation test using the PEP tool [11] in which we initialised our model to a state representing the exponential growth phase but then activated the *Signal* entity to represent the presence of carbon stress. The sequence of steps resulting from this simulation are shown in Figure 6; the first column represents the initial state and each subsequent column represents the next observed state. The results show that our model correctly switches growth phases by entering a strong attractor cycle of period two that correctly represents the physiological conditions present in the stationary growth phase [14]. In particular, we see as expected that the level of stable RNA and *Fis* decline to very low levels with the increasing concentration of CRP [25].

<i>RRN</i>	1	1	1	0	0	0	0	0	0
<i>Crp</i>	1	1	1	1	1	2	3	3	3
<i>Cya</i>	3	3	3	3	3	3	3	3	3
<i>TopA</i>	0	0	0	0	0	0	0	0	0
<i>Fis</i>	4	3	2	1	0	0	0	0	0
<i>Signal</i>	1	1	1	1	1	1	1	1	1
<i>GyrAB</i>	2	3	2	3	2	3	2	3	2

Figure 6: Simulating the switch from exponential to stationary phase in *E. coli*.

Further validation was successfully performed on the model, for example to check that the model correctly returns to the exponential growth phase [25] when Signal is inactive.

### (ii) Dynamic Properties

Next we consider using our model to investigate experimental hypotheses which we formulate using the insights gained from the analysis so far and from the experimental literature. We make use of powerful model-checking techniques and in particular, use the PEP extended reachability tool [11] which allows the reachability of a system state to be checked.

As an example, it can be seen from the literature that the level of stable RNA and Fis should remain low when Signal is active [25]. We checked this by setting the initial state of our model such that Signal is active (state 1), stable RNA is inactive (state 0) and Fis is low (0 or 1). We then use PEP to correctly confirm that no state is reachable satisfying the constraint  $RRN > 0$  (i.e. in which stable RNA is active). A similar check can be performed to see if during the stationary phase entities TopA and GyrAB can both become inactive (i.e. simultaneously have state 0). Interestingly, it turns out that PEP is able to find a reachable state in which this occurs, despite the literature indicating that the entities are mutually exclusive [25]. The model checker returns a firing trace which leads to a witness state and this can be automatically simulated using PEP to provide insight into how this behaviour can occur.

### (iii) Mutant Analysis

The final step in our analysis was to investigate the affect of “fixing” an entity in the model to an explicit state. This corresponds to the experimental approach of creating mutants in which genes are knocked out or overexpressed, and provides a means of investigating the robustness of the network when key components do not function as normal. The idea is to ignore the state transition table for the entity in question so that it becomes an input entity like Signal.

We investigated the affect that knocking out and overexpressing the entities *crp*, *cya*, *gyrAB* and *topA* had on the production of Fis and consequently the expression of the *rrn* operon. In particular, we considered two scenarios: firstly, in the absence of carbon stress (i.e. Signal is inactive) can stable RNA be prevented from being expressed; and secondly, in the presence of carbon stress (i.e. Signal is active) can stable RNA be expressed. We performed these tests by first setting Signal, Fis and stable RNA to be inactive (state 0), and then knocking out and overexpressing the remaining entities in turn. We then repeated the analysis with Signal set to active. The observed results of this mutant analysis are summarised in Figure 7.

Entity	KO	OE	KO (s)	OE (s)
CRP	Yes	Yes	Yes	No
Cya	Yes	Yes	Yes	No
GyrAB	No	Yes	No	No
TopA	Yes	No	No	No

**Figure 7: Results of knocking out (KO) and overexpressing (OE) entities (where (s) denotes the presence of carbon stress).**

When Signal is inactive, we notice that knockout and overexpression of *crp* and *cya* allows for the production of stable RNA. However, knocking out and overexpressing *gyrAB* and *topA* respectively does not. In the case of knocking out *gyrAB*, a low concentration of GyrAB prevents an increase in negative DNA supercoiling; thus Fis production is reduced and so stable RNA production is low [25]. Overexpression of *topA* inhibits the amount of negative DNA supercoiling, thus reducing Fis production and therefore stable RNA production [25].



When Signal is active, we notice different behaviour; knocking out both *crp* and *cya* allows for the production of stable RNA even under carbon stress. This is due to the reduced activation of the implicit complex cAMP.CRP, which in turn does not repress *fis* so strongly, and thus stable RNA is allowed to accumulate. However, when we overexpress *crp* and *cya*, the opposite occurs and stable RNA is not produced as expected [25].

## 6 Model Comparison

We now compare and contrast our multi-valued model to a corresponding Boolean model [29] which was constructed using the same data source [25]. On inspection, both models of the carbon stress response in *E. coli* appear to capture similar fundamental behaviour: the switch from the exponential growth phase to the stationary phase (and vice versa); and the mutual inhibition between *fis* and *crp* [25]. However, an interesting observation is that TopA erroneously reaches an activated level in the Boolean model during the switch from the exponential to the stationary growth phase. In the multi-valued model, the level of TopA correctly remains low, allowing the amount of DNA supercoiling to increase [25]. In fact, the only time when the level of TopA can temporarily rise to an intermediate level of 1 in the multi-valued model is when it is currently at state 0 and Fis and GyrAB are fully active [25]; TopA then immediately drops back to 0. Under the Boolean abstraction, the conditions for TopA to be low and Fis and GyrAB to be high also hold, but the ability for TopA to increase to an intermediate state is not possible, leading to unrealistic results.

Another interesting comparison between the models concerns the results of mutant analysis. We performed the same mutant analysis on the entities *crp*, *cya*, *gyrAB* and *topA* in the Boolean model as in Figure 7 to compare how the production of stable RNA is affected. The results of this analysis are shown in Figure 8, where the discrepancies between the models are highlighted in brackets. The analysis shows that in a number of cases stable RNA reaches an activated level

Entity	KO	OE	KO (s)	OE (s)
CRP	Yes	Yes	Yes	(Yes)
Cya	Yes	Yes	Yes	(Yes)
GyrAB	No	Yes	No	(Yes)
TopA	Yes	No	(Yes)	No

Figure 8: Results of entity knock out (KO) and overexpression (OE) in the Boolean model.

(i.e. state 1) in the Boolean mutants which contradicts the results for the multi-valued model. This appears to be due to the Boolean model being less restrictive with respect to the activation of Fis.

Clearly, the Boolean model fails to capture some subtle aspects of the genetic network's behaviour. This raises a number of interesting questions concerning the relationship between these two qualitative modelling approaches, such as what it means for the behaviour of a multi-valued model to be representable in the Boolean domain. The answers to these questions have real practical relevance, since the ability to refine a model into a simpler one eases user-comprehension and allows a reduction in the models state space, providing important advantages for computation and analysis purposes.

To begin to address the above questions, we formalize the idea of a Boolean model  $B$  faithfully representing the behaviour of a multi-valued model  $MV$ . We view the behaviour of any model

$X$  as the set of observable state traces  $T(X)$ , where a trace is the sequence of states resulting from a simulation of the model starting from a global initial state. To view the behaviour of  $MV$  in the Boolean domain, we introduce a *Boolean state mapping*:

$$\phi = \langle \phi_{g_i} : S_{g_i} \mapsto \{0, 1\} \mid g_i \in MV \rangle$$

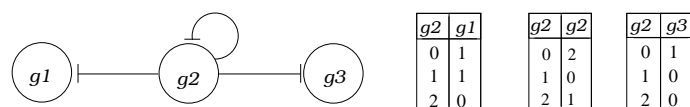
a family of surjective mappings which map the state space  $S_{g_i}$  of each entity  $g_i$  of  $MV$  to the Boolean states 0 and 1. We can apply  $\phi$  to  $T(MV)$  to translate the traces to the Boolean domain resulting in the traces denoted by  $\phi(T(MV))$ . We can now formalise the concept of  $B$  being a *Boolean refinement* of  $MV$  as follows.

**Definition 1** (Boolean Refinement). *Let  $MV$  be a multi-valued model,  $B$  be a Boolean model with the same structure as  $MV$  and  $\phi$  be a Boolean state mapping for  $MV$ . A Boolean refinement is a pair  $(B, \phi)$  such that  $T(B) \subseteq \phi(T(MV))$ .*

Intuitively, the definition of a Boolean refinement says that the behaviour of  $B$  must be contained within the behaviour of  $MV$  under  $\phi$ . An important question is therefore whether we can always find a Boolean refinement for a multi-valued model and this is answered by the following theorem.

**Theorem 1.** *Not every multi-valued model  $MV$  has a Boolean refinement  $(B, \phi)$ .*

*Proof.* To prove this theorem, we need only find a single multi-valued model which does not have a Boolean refinement. Consider Figure 9 which illustrates a simple multi-valued model  $MV$  with three entities. The idea is that  $g_2$  inhibits the other entities (including itself); it inhibits  $g_1$  when in state 2 and  $g_3$  when in state 1 or above. Since  $g_1$  and  $g_3$  are Boolean entities,



**Figure 9:** Multi-valued model  $MV$  with three entities  $g_1$ ,  $g_2$  and  $g_3$ .

there are two possibilities for  $\phi_{g_1}$  and  $\phi_{g_3}$ . Furthermore,  $g_2$  has three states and therefore six possible mappings exist for  $\phi_{g_2}$ . This is compounded by the fact that for each of the 24 (i.e.  $2 * 6 * 2$ ) Boolean state mappings  $\phi$  we have to consider 256 possible Boolean models in an exhaustive search. To overcome this prohibitively large search space, we can limit the models that need to be considered by making use of the following facts: i) isomorphic mappings can be ignored (leaving only three mappings for  $\phi_{g_2}$  and one for  $\phi_{g_1}$  and  $\phi_{g_3}$ ), and; ii) for each  $\phi$ , we can translate the state transition tables of  $MV$  into the Boolean tables  $\phi(MV)$  to restrict the set of Boolean models to be considered. Taking this approach limits the total number of possible Boolean refinements to just 16; we have 4 for  $\phi_2 = \{0 \mapsto 0, 1 \mapsto 0, 2 \mapsto 1\}$ , 4 for  $\phi_2 = \{0 \mapsto 0, 1 \mapsto 1, 2 \mapsto 1\}$  and 8 for  $\phi_2 = \{0 \mapsto 0, 1 \mapsto 1, 2 \mapsto 0\}$ . We are able to show that all these Boolean models fail to exhibit behaviour consistent with  $\phi(T(MV))$ . Note for brevity the details of these checks are omitted here.  $\square$

The above result is important as it provides clear motivation for the use of multi-valued modelling techniques and represents a first step in understanding the relationship between multi-valued and Boolean models. Research is now ongoing to investigate the existence and construction of Boolean refinements.

## 7 Conclusions

In this paper, we have developed a formal framework for modelling and analysing multi-valued models of genetic regulatory networks based on using high-level Petri nets. This extends the Boolean approach of [29], providing important analysis support for the more realistic multi-valued modelling approach. We see the key contributions of this paper as follows: i) Extending Petri net models of Boolean networks [29] to multi-valued models; ii) A new systematic process for constructing multi-valued models of genetic regulatory networks based on logic minimization; iii) Tool support to automate model construction; iv) A detailed case study which explores the application of Petri net techniques [7, 17] to analysing a multi-valued model of a genetic network; v) An initial investigation into the formal relationship between Boolean and multi-valued models.

Work is now ongoing to further investigate the notion of model refinement with an emphasis on results concerning the existence and automated construction of refinements. Such results appear to be important if formal techniques like ours are to be able to cope with the challenges of modelling large, realistic biological networks. We are also interested in exploring the idea of compositional model construction and analysis in this context. A compositional approach to model construction would be consistent with the biological intuition that genetic networks are modular [32] and could lead to efficient analysis techniques. In particular, we see compositional model checking techniques [3] as an important area of research here.

## Acknowledgement

We are very grateful to A. Wipat, M. Koutny, V. Khomenko and the anonymous referees for useful comments on this work. We thank the EPSRC for supporting R. Banks, the BBSRC for support via CISBAN, and the Newcastle Systems Biology Resource Centre.

## References

- [1] T. Akutsu, S. Miyano and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. *Proc. of Pac. Symp. on Biocomputing*, 4:17-28, 1999.
- [2] E. Best, H. Fleischhack, W. Fraczak, R. P. Hopkins, H. Klaudel and E. Pelz. A class of composable high level Petri nets. *ICATP'95*, LNCS 935, pages 103–120, Springer, 1995.
- [3] T. Bultan, J. Fischer and R. Gerber. Compositional verification by model checking for counter-examples. *Int. Symp. on Software Testing and Analysis*, 21(3): 224-238, 1996.
- [4] C. Chaouiya, E. Remy, P. Ruet and D. Thiéffry. Qualitative modelling of genetic networks: From logical regulatory graphs to standard petri nets. *ICATP'2004*, LNCS 3099, pages 137–156. Springer, 2004.
- [5] J. Comet, H. Klaudel and S. Liazu. Modeling multi-valued genetic regulatory networks using high-level Petri nets. *ICATP'2005*, LNCS 3536, pages 208–227, Springer, 2005.
- [6] P. D'haeseleer, S. Liang and R. Somogyi. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707-726, 2000.
- [7] J. Esparza. Model checking using net unfoldings. *Science of Computer Programming*, 23(2-3):151–195, 1994.

- [8] H.J. Genrich and K. Lautenbach. System Modelling with High-Level Petri Nets. *Theor. Comp. Sci.*, 13(1):109–136, 1981.
- [9] C. Gershenson. Classification of random boolean networks. In: R. K. Standish et al (eds), *Proc. of the 8th Int. Conf. on Artificial Life*, pages 1–8, MIT Press, 2002.
- [10] GNaPN website, <http://bioinf.ncl.ac.uk/gnapn/>, 2007.
- [11] B. Grahlmann. The pep tool. *Computer Aided Verification*, LNCS 1254, pages 440–443, Springer, 1997.
- [12] P. Grossman. *Discrete Mathematics for Computing*. Palgrave MacMillan, 2nd Ed., 2002.
- [13] M. Heiner and I. Koch. Petri Net Based Model Validation in Systems Biology. *ICATP'2004*, LNCS 3099, pages 216–237. Springer, 2004.
- [14] R. Hengge-Aronis. The general stress response in *Escherichia coli*. Bacterial Stress Responses, *American Society for Microbiology Press*, pages 161–178, 2000.
- [15] K. Jensen. Coloured Petri Nets - basic concepts, analysis methods and practical use. *EATCS Monographs on Theoretical Computer Science 1*, Springer, 1992.
- [16] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:437–467, 1969.
- [17] V. Khomenko. *Model Checking based on Prefixes of Petri Net Unfoldings*. PhD thesis, School of Computing Science, University of Newcastle upon Tyne, UK, 2003.
- [18] H. Matsuno, C. Li and S. Miyano. Petri Net Based Descriptions for Systematic Understanding of Biological Pathways. *IEICE Trans. Fundam.*, E89-A(11):3166–3174, 2006.
- [19] A. Mishchenko and R. Brayton. Simplification of non-deterministic multi-valued networks. In *Proc. ICCAD*, 2:557–562, ACM Press, 2002.
- [20] T. Murata. Petri nets: Properties, analysis and applications. *Proc. of the IEEE*, 77(4):541–580, 1989.
- [21] J. Peterson. Petri nets. *Computing Surveys*, 9(3):223–252, ACM Press, 1977.
- [22] Petri Net World, <http://www.informatik.uni-hamburg.de/TGI/PetriNets>, 2007.
- [23] V. Reddy, M. Liebman and M.L. Mavrovouniotis. Qualitative analysis of biochemical reaction systems. *Computers in Biology and Medicine*, 26(1):9–24, 1996.
- [24] W. Reisig and G. Rozenberg. *Lectures on Petri Nets I: Basic Models*, Advances in Petri nets, LNCS 1491, Springer, 1998.
- [25] D. Ropers, H. De Jong, M. Page, D. Schneider and J. Geiselmann. Qualitative simulation of the carbon starvation response in *Escherichia coli*. *Bio. Systems*, 84(2):124–152, 2006.
- [26] R. Rudell and A. Sangiovanni-Vincentelli. Multiple-valued minimization for pla optimization. *IEEE Transactions on Computer-Aided Design*, 6(5):727–750, 1987.
- [27] A. Silvescu and V. Honavar. Temporal boolean network models of genetic networks and their inference from gene expression time series. *Complex Systems*, 13(1):54–70, 2001.
- [28] L. J. Steggles, R. Banks, O. Shaw and A. Wipat. Qualitatively modelling and analysing genetic regulatory networks: a Petri net approach. *Bioinformatics*, 23(3):336–345, 2006.
- [29] L. J. Steggles, R. Banks and A. Wipat. Modelling and analysing genetic networks: From Boolean networks to Petri nets. *CMSB'06*, LNCS 4210, pages 127–141. Springer, 2006.
- [30] R. Thomas. Kinetic logic: A Boolean approach to the analysis of complex regulatory systems. *Notes Biomath*, 29, 1979.
- [31] R. Thomas. Regulatory networks seen as asynchronous automata: A logical description. *Theor. Biol.*, 153: 1–23, 1991.
- [32] A. Wagner. *Robustness and Evolvability in Living Systems*. Princeton Univ. Press, 2005.

## Supplementary Material

This document provides a detailed listing of the *E.coli* carbon starvation model. The model consists of 6 entities and an additional input entity Signal. These entities are listed below along with their state space:

- $RRN\{0,1\}$ ;
- $Crp\{0,1,2,3\}$ ;
- $Cya\{0,1,2,3\}$ ;
- $Fis\{0,1,2,3,4,5\}$ ;
- $GyrAB\{0,1,2,3\}$ ;
- $TopA\{0,1,2,3\}$ ;
- $Signal\{0,1\}$ .

A complete listing of the equations describing the dynamics of this model is provided below:

$$RRN\{0\} = Fis\{0,1\} + Fis\{0,2\}$$

$$RRN\{1\} = Fis\{3,4,5\}$$

$$Crp\{1\} = Crp\{0\} + Crp\{0,1\} Fis\{2,3,4,5\} + Crp\{0,2\} Fis\{2,3,4,5\} + Crp\{0,1\} Fis\{1,4,5\} + Crp\{0,2\} Fis\{1,4,5\}$$

$$Crp\{2\} = Crp\{3\} Fis\{1,2,3,4,5\} + Crp\{1\} Fis\{0\}$$

$$Crp\{3\} = Crp\{2,3\} Fis\{0\}$$

$$Cya\{1\} = Cya\{0\}$$

$$Cya\{2\} = Cya\{1\} + Crp\{3\} Cya\{1,3\} + Signal\{1\}$$

$$Cya\{3\} = Cya\{2,3\} Signal\{0\} + Cya\{2\} + Crp\{0,1\} Cya\{2,3\} + Crp\{0,2\} Cya\{2,3\}$$

$$Fis\{0\} = Crp\{1,2,3\} Cya\{1,2,3\} Fis\{0,1\} Signal\{1\}$$

$$Fis\{1\} = Fis\{0\} Signal\{0\} + Fis\{0,1\} Signal\{0\} TopA\{2,3\} + Fis\{2\} TopA\{2,3\} + Fis\{0,1\} GyrAB\{0,1\} Signal\{0\} + Fis\{2\} GyrAB\{0,1\} + Cya\{0\} Fis\{0\} + Crp\{0\} Fis\{0\} + Cya\{0\} Fis\{0,1\} TopA\{2,3\} + Crp\{0\} Fis\{0,1\} TopA\{2,3\} + Cya\{0\} Fis\{0,1\} GyrAB\{0,1\} + Crp\{0\} Fis\{0,1\} GyrAB\{0,1\} + Crp\{1,3\} Cya\{1,3\} Fis\{2\} Signal\{1\} + Crp\{2,3\} Cya\{1,3\} Fis\{2\} Signal\{1\} + Crp\{1,3\} Cya\{2,3\} Fis\{2\} Signal\{1\} + Crp\{2,3\} Cya\{2,3\} Fis\{2\} Signal\{1\}$$

$$Fis\{2\} = Fis\{3\} GyrAB\{0,1\} + Crp\{1,2,3\} Cya\{1,2,3\} Fis\{3\} Signal\{1\} + Fis\{3\} TopA\{2,3\} + Crp\{0\} Fis\{1\} GyrAB\{2,3\} TopA\{0,1\} + Cya\{0\} Fis\{1\} GyrAB\{2,3\} TopA\{0,1\} + Fis\{1\} GyrAB\{2,3\} Signal\{0\} TopA\{0,1\}$$

$$Fis\{3\} = Fis\{4\} GyrAB\{0,1\} + Crp\{1,2,3\} Cya\{1,2,3\} Fis\{4\} Signal\{1\} + Fis\{4\} TopA\{2,3\} + Crp\{0\} Fis\{2\} GyrAB\{2,3\} TopA\{0,1\} + Cya\{0\} Fis\{2\} GyrAB\{2,3\} TopA\{0,1\} + Fis\{2\} GyrAB\{2,3\} Signal\{0\} TopA\{0,1\}$$

$$Fis\{4\} = Fis\{5\} + Crp\{0\} Fis\{3,5\} GyrAB\{2,3\} TopA\{0,1\} + Cya\{0\} Fis\{3,5\} GyrAB\{2,3\} TopA\{0,1\} + Fis\{3,5\} GyrAB\{2,3\} Signal\{0\} TopA\{0,1\}$$

$$\text{Fis}\{5\} = \text{Crp}\{0\} \text{Fis}\{4\} \text{GyrAB}\{2,3\} \text{TopA}\{0,1\} + \text{Cya}\{0\} \text{Fis}\{4\} \text{GyrAB}\{2,3\} \text{TopA}\{0,1\} \\ + \text{Fis}\{4\} \text{GyrAB}\{2,3\} \text{Signal}\{0\} \text{TopA}\{0,1\}$$

$$\text{GyrAB}\{0\} = \text{Fis}\{4,5\} \text{GyrAB}\{0,1\}$$

$$\text{GyrAB}\{1\} = \text{Fis}\{4,5\} \text{GyrAB}\{2\} + \text{Fis}\{0,1,2,3\} \text{GyrAB}\{0\}$$

$$\text{GyrAB}\{2\} = \text{GyrAB}\{3\} \text{TopA}\{0\} + \text{Fis}\{0,2,3\} \text{GyrAB}\{1\} + \text{Fis}\{0,1\} \text{GyrAB}\{1\} \\ + \text{Fis}\{4,5\} \text{GyrAB}\{3\}$$

$$\text{GyrAB}\{3\} = \text{Fis}\{0,1,2,3\} \text{GyrAB}\{2\} + \text{Fis}\{0,1,2,3\} \text{GyrAB}\{3\} \text{TopA}\{1,2,3\}$$

$$\text{TopA}\{0\} = \text{TopA}\{1\} + \text{GyrAB}\{0,1\} \text{TopA}\{0,1\} + \text{GyrAB}\{0,1\} \text{TopA}\{0,1\} + \text{Fis}\{0,2,3\} \text{TopA}\{0,1\} \\ + \text{Fis}\{0,1\} \text{TopA}\{0,1\}$$

$$\text{TopA}\{1\} = \text{Fis}\{4,5\} \text{GyrAB}\{2,3\} \text{TopA}\{0,2\} + \text{TopA}\{2\}$$

$$\text{TopA}\{2\} = \text{TopA}\{3\}.$$

The state transition tables and Petri net for this model can be downloaded by visiting our project website at <http://bioinf.ncl.ac.uk/gnapn>.