# Evolving Concurrent Petri Net Models of Epistasis

Michael Mayo[1], Lorenzo Beretta[2]

[1]Dept. of Computer Science, University of Waikato, New Zealand
[2]Referral Center for Systemic Autoimmune Diseases, Fondazione IRCCS Ospedale Maggiore Policlinico di Milano, Italy
[1]mmayo@cs.waikato.ac.nz, [2]lorberimm@hotmail.com

**Abstract.** A genetic algorithm is used to learn a non-deterministic Petri net-based model of non-linear gene interactions, or statistical epistasis. Petri nets are computational models of concurrent processes. However, often certain global assumptions (e.g. transition priorities) are required in order to convert a non-deterministic Petri net into a simpler deterministic model for easier analysis and evaluation. We show, by converting a Petri net into a set of state trees, that it is possible to both retain Petri net non-determinism (i.e. allowing local interactions only, thereby making the model more realistic), whilst also learning useful Petri nets with practical applications. Our Petri nets produce predictions of genetic disease risk assessments derived from clinical data that match with over 92% accuracy.

**Keywords:** Petri net, genetic algorithm, epistasis, concurrency, systemic schlerosis, digital ulcers.

## 1. Introduction

Petri nets [13] are widely used abstract computational models of concurrent processes. Recently, they have found application as useful modeling tools in biochemistry, genetics and medicine (e.g. [2,6]).

They are best described as executable graphs with two different types of node: places and transitions. In a biochemical modeling situation, a place usually represents a substance and a transition stands for a reaction or process in which one or more input substances are transformed over time into one or more output substances. Petri nets have potential to realistically model what could be happening in real world situations because they are inherently concurrent. For example, in a net, two pathways of multiple transitions may fire simultaneously, thus simulating two concurrent processes.

Figure 1 depicts a simple Petri net with three places and two transitions. The places, $P_0$, $P_1$ and $P_2$, represent three different chemical substances, and the transitions, $T_0$ and $T_1$, represent two different reactions that can occur between them. Petri nets represent the concentration of a substance at a particular point in time by

"marking" each place with an integer number of tokens. These tokens move around the net as the transitions fire.

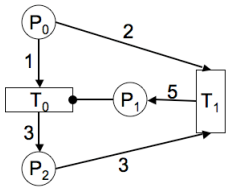


**Fig. 1.** Example of a Petri Net.

For example, suppose in Figure 1 that there are 10 tokens at $P_0$, and no tokens at $P_1$ and $P_2$. The overall marking of the entire net is the vector <10,0,0>. The arcs indicate either transition inputs or outputs, depending on the directionality. They are labeled with a quantity of tokens consumed or produced. $T_0$, for example, represents a chemical process in which $P_0$ is being converted into $P_2$, with one unit of $P_0$ being consumed for every three units of $P_2$ being produced. If $T_0$ fires once, the marking of the net will become <9,0,3>. If it fires twice, it will become <8,0,6>. $T_1$, on the other hand, represents an entirely different reaction with $P_0$ and $P_2$ as inputs, and $P_1$ as output. Because $T_1$ requires three units of $P_2$ as an input, it cannot fire until $T_0$ has fired at least once. If this happens, the marking will change from <10,0,0> to <9,0,3> (after $T_0$ fires) and then to <7,5,0> (after $T_1$ fires).

Transitions can only fire if there are sufficient input tokens available (i.e. the number of tokens at an input place cannot fall below zero), and if they are not inhibited. An example of an inhibitor in Figure 1 is the arc from $P_1$ to $T_0$: if ever $P_1$ has a non-zero quantity of tokens present, then $T_0$ is effectively turned off.

The only other time that a transition cannot fire is if one of its output places has insufficient capacity. For example, suppose the maximum capacity of all places in Figure 1 is 10 tokens, and the current marking is <7,6,9>. Although $T_1$ has sufficient inputs available at $P_0$ and $P_2$, there is insufficient capacity at the output place $P_2$, so $T_1$ cannot fire.

It should be evident by now that Petri nets are concurrent and non-deterministic models. Transitions may fire in any order, and if they do not share common inputs or outputs, they can fire concurrently.

Non-determinism does have some issues when models are to be executed on serial computers. If there are two or more transitions enabled, which one should fire first? The simplest answer to this question is to enforce an arbitrary priority amongst the transitions [13]. For example, in Figure 1, $T_0$ may have a higher priority and therefore always fire before $T_1$, if they are both enabled at the same time. This strategy simplifies a non-deterministic Petri net into a deterministic model.

An alternative answer is to make the transitions fire stochastically. Of those that are enabled, one of them is selected to fire at random; and sometimes, in order to give all enabled transitions a fair chance of firing, those that have recently fired are not permitted to fire again until a certain amount of time has elapsed.

A significant issue with both of these solutions is that they require global coordination. In other words, in order to select the next transition to fire, all transitions must be examined globally. Nature, however, is unlikely to employ this level of global coordination; natural systems are more likely to evolve gradually with many local, concurrent interactions. The issue is therefore how to relax the requirement of global coordination from our Petri net models in order to make them more realistic and therefore more interesting.

In this paper, we address this specific problem in the context of modeling disease-causing epistatic interactions between genes. Our solution is to convert the Petri net

model into an alternative representation called a set of state trees, which represents all possible orderings in which the transitions can fire. The leaves of these trees therefore represent all possible final outcomes.

We show that it is possible to evolve a Petri net using a genetic algorithm whose state tree outcomes match clinical observations in over 92% of the outcomes. Furthermore, it is also possible to limit the depth and size of the trees so that the tree remains relatively small, thereby permitting inspection.

This new approach eliminates the need for global coordination of the transition firings in the Petri net. Instead, transitions can fire in any order, and the Petri net therefore exhibits only more realistic local interactions.

## 2. Method

We describe firstly our Petri net models of non-linear gene interaction, and then discuss the conversion of a Petri net to a multiple state trees. Finally we describe the specific genetic algorithm that we employed to learn our Petri net-based models.

### 2.1 Petri Net Models of Epistasis

Epistasis [11] refers to the phenomenon of non-linear gene interaction. In the context of genetic disease, it manifests when no single genetic cause for a disease can be isolated; instead, scientists determine that it is the curious interaction between multiple genes that causes the disease. The main question is how this interaction could be happening, and Petri nets are useful as a means of hinting at a hypothesis explaining the interaction.

In biological reality, each gene is actually a sequence comprising hundreds of thousands of nucleotides. Mutations to these sequences may occur in many ways, but one of the most common is a change to a single nucleotide, known as a Single Nucleotide Polymorphism (SNP). A single SNP may completely alter the behavior of a gene. In this paper, we will refer to the value of an SNP as *A* (the original, wild-type) or *a* (its mutant form). In an individual, nucleotides come in unordered pairs (alleles); so therefore an individual has three possible genotypes per SNP: *AA*, *Aa*, or *aa*[1].

For modeling purpose, the nucleotide level of detail is far too complex. We therefore model entire genes as "gene units" within our Petri nets. Each gene unit is assumed to vary *only* by a single SNP; that is, all nucleotides except for one are assumed constant. This representation is depicted in Figure 2.

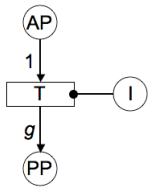


**Fig. 2.** A gene unit.

As Figure 2 illustrates, a gene is modeled as two places and a transition. The first place is called the "activating place" (AP) and represents the substance that activates or turns on the gene; the second substance is the "product place" (PP), and represents

[1] This is a convention we use in this paper for readability by non-geneticists. To be technically correct, we should use nucleotide notation, e.g. CC/CG/GG.

the output of the gene. There is also an optional "inhibitory place" (I) that can turn the gene unit off completely. The key point is that the rate of production of the gene unit, the value *g*, is controlled by a genotype varying only by a single SNP.

Following biological investigations [3,5,12], it is assumed that the SNP's mutant form *a* causes an over-production of the gene's output substance at some fixed ratio. The values of *g* in Figure 2, therefore, have been set to 3 for genotype *AA*; 6 for genotype *Aa*; and 9 for genotype *aa*.

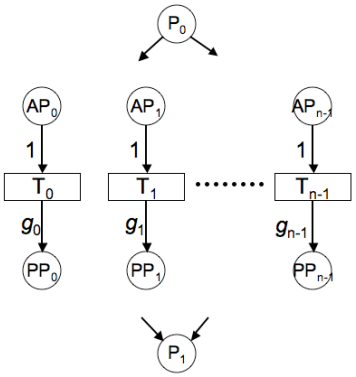


**Fig 3.** A generalized Petri net architecture comprising *n* gene units.

Figure 3 shows an overall Petri net-based architecture comprising several gene units. It should evident that whenever *n* genes are being modeled, there must be up to $3^n$ different genotypes involved.

There are two additional places in this architecture: $P_0$ and $P_1$. $P_0$ denotes the initial source of tokens in the network, or from a biological point of view, it is the trigger event that initiates the chain of reactions leading up to the disease. $P_1$ represents the output of this process; it is the toxic disease-causing substance. Following previous studies [1,7,9], we use a threshold to determine whether the toxic substance is in such abundance as to cause a high risk of the disease. In all of our simulations, this threshold is set to 50% of the maximum capacity of $P_1$. Thus, if the threshold at $P_1$ is reached or exceeded, it is assumed that the current genotype leads to a high risk of disease; otherwise, there is only a low risk.

Besides the gene units and $P_0$ and $P_1$, we also assume the existence of an arbitrary additional number of places and transitions. These are places and transitions not forming the parts of any specific gene unit, but they do have significant influence because they connect to the gene unit's APs and PPs.

In all of our simulations the maximum place capacity and arc weights are set to 10.

### 2.2 From Petri Nets To Sets Of State Trees

Petri nets are inherently non-deterministic, concurrent computational models. That is, transitions that are co-enabled can fire in any order, as long as one of the transitions does not disable the other, and transitions that do not share common inputs and outputs may fire concurrently. In order to evaluate the behaviour of such a model it is necessary to "unroll" its non-deterministic aspects into a deterministic form that can be properly assessed.

We propose a tree representation that we call a state tree as the deterministic form of a Petri net. A state tree is an alternative representation of a Petri net in which nodes represent markings, and arcs represent transitions. A path from the root of the state tree to a leaf represents, therefore, a single execution of the Petri net from start state to

final state. Figure 4 depicts a state tree for the very simple Petri net depicted in Figure 1.

In Figure 4, the starting state is <10,0,0>, indicating 10 tokens at $P_0$ and no tokens anywhere else. Only transition $T_0$ is enabled initially, but after it fires once, both $T_0$ and $T_1$ are thereafter enabled. The final states that are reached, which depend on the ordering of transition firings, are <7,0,9>, <6,5,3>, or <7,5,0>.

Clearly, for a Petri net of significant size or complexity, the state tree can be very large. Furthermore, if a state is visited more than once, then at that point the state tree can have effectively infinite depth. To resolve these problems, we limited the depth of our state trees to 10 and automatically excluded from consideration any Petri nets whose state tree exceeded this depth limit. We also limited the number of leaves per tree to 100 or less, again excluding from consideration any trees that did not conform. Finally, we also made use of a domain-specific heuristic to further trim the tree. Since $P_1$ only ever accumulates tokens and is never the input place for another transition, it is possible to stop growing the state tree as soon as the number of tokens at $P_1$ exceeds the threshold of 50%, since the risk assessment will thereafter not change.

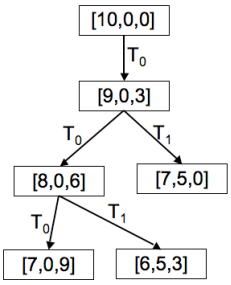


**Fig 4.** A state tree for the Petri net depicted in Figure 1.

These measures for the most part kept the size of the state trees manageable, whilst still being practical for solving the problem of non-linear gene modeling.

As Figure 3 shows, there are $n$ variables $g_0$, $g_1$, …, $g_{n-1}$, that are genotype dependent within each net. As each gene has 3 different possible values (*AA*/3, *Aa*/6 or *aa*/9), this means that there are $3^n$ possible genotypes. Now, each genotype will produce a different Petri net execution dynamics, and therefore a different state tree must be constructed for each and every genotype. Thus, in our problem domain, every single Petri net is converted into not one but a set of $3^n$ state trees.

### 2.3 Evolving Petri Nets

We propose the use of a genetic algorithm to learn a Petri net model of the observed non-linear gene interactions. Genetic algorithms [4] use random mutations and crossover operators to gradually optimize solutions to problems. In the specific field of gene interaction modeling, Moore and Hahn [9], Mayo [7], Mayo and Beretta [14], and Beretta et al. [1] all apply genetic algorithms to learn Petri nets. The key difference between those previous works and our current work presented here is that previously, deterministic Petri nets were used, whereas now we are concerned with relaxing the determinism criteria and instead learning Petri nets that may execute non-deterministically (i.e. the transitions may fire in any order) whilst still remaining a highly accurate model of the interaction.

In our case, we have a set of $3^n$ genotypes, each genotype being labeled either “high risk” or “low risk”, and we want a Petri net that, after all transitions have fired, always reaches or exceeds the threshold at $P_1$ for high risk genotypes, but never

exceeds the threshold at $P_1$ for low risk genotypes. Construction of a state tree for each genotype, therefore, is essential in order to assess all possible outcomes. The model should show how the genes activate, produce, and interact in all situations in order to produce the correct desired behavior.

Our representation of a Petri net for the genetic algorithm is as follows: we fix the number of places to $2n+2$ and the number of transitions to $n+10$, where $n$ is the number of genes, and model each net as a list of directed arcs. Arcs can be either weighted or inhibitory. Our genetic algorithm randomly constructs its initial Petri nets, putting a random arc with random weight between a place and transition with probability 0.2. Of those arcs, 10% of them are chosen randomly to be inhibitors.

Our genetic algorithm has a population size of 2,000 individuals. From the random initial population and for each subsequent generation, the top 5% of individuals are retained for the following generation. The rest are created via either the mutation operator or the crossover operator. The mutation operator either (i) adds one, two or three random arcs to the net; or, (ii) deletes a random arc, or (iii) modifies an existing arc by changing its weight or type, with equal probability. The crossover operator merges the arc lists of two parent nets, while maintaining the criteria that there is no more than one arc between any pair of nodes.

In our initial testing, we found that the mutation operator was far more effective than the crossover operator, and so set the probability of crossover to 5% and the probability of mutation to 95%. Parent nets are selected stochastically with probability proportionate to fitness. The genetic algorithm continues to iterate until 2,000 generations pass without any gains in fitness. At that point, the search is complete and the best net is returned.

In our non-linear gene modeling scenario, there are $3^n$ genotypes, and therefore $3^n$ state trees per net. To compute the fitness of each net, we iterate over the genotypes and generate for each genotype its corresponding state tree. For example, if $n$=3, then the genotypes will be *AA-AA-AA*, *AA-AA-Aa*, *AA-AA-aa*, …, *aa-aa-aa*, where *AA* corresponds to arc weight 9, *Aa* to weight 6, and *aa* to weight 3.

Since each genotype will have a risk assessment (either high or low), we examine the leaves of its state tree and compute the proportion of leaves with the correct predicted assessment. This is what we term the accuracy of the state tree. The overall fitness is then the average accuracy across all genotypes, with a small bias against net size subtracted. During testing, we also found that squaring this fitness value tended to give marginally better results than not squaring it, and so the final result is squared. In mathematical terms, the fitness function is given by the equation below, where $r$ is a genotype.

$$fitness(net) = \left[ \frac{\sum_{r=0}^{3^n-1} accuracy(net, r)}{3^n} - 0.01 \times size(net) \right]^2$$

The fitness function ranges in value between 0.0 and 1.0, with a greater value indicating a better solution. The size component of the function is determined by dividing the actual number of arcs by the maximum possible number.

Figure 5 illustrates the computation of the fitness value for the very simple Petri net from Figure 1, assuming that the $P_0$-$T_0$-$P_2$ portion of the net is now a single gene unit. Since $g$ can take three possible values, specifically 3,6 or 9, there are three possible state trees. If the *AA* genotype is low risk ($P_1$ must be less than 50% of maximum capacity) whilst the *Aa* and *aa* genotypes are high risk ($P_1$ must be greater than or equal to 50% capacity, which is 5 tokens), then Figure 5 shows that this net is only 33.3% accurate when $g$=3, but 100% accurate when $g$=6 or 9. Overall, then, the fitness of this net is (0.33+1.0+1.0)/3.0-0.01(6/6)≈0.77

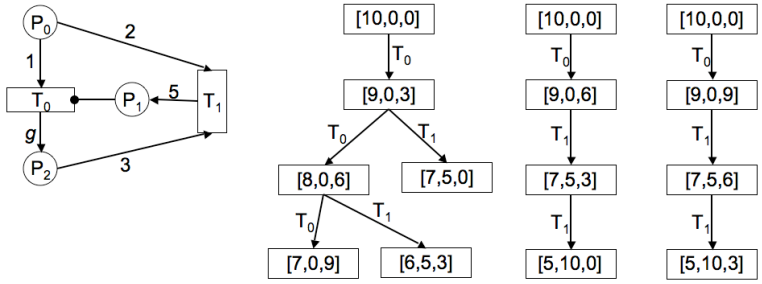


**Fig. 5.** (a) A Petri net with a single gene unit and (b) its corresponding state trees for $g$=3, 6 and 9 respectively.

## 3. Evaluation

### 3.1 Non-Linear Gene Interaction Model of Digital Ulcers

A recently discovered disease-causing non-linear gene interaction is used as a test-bed for our method [1]. This model, depicted in Figure 6, describes the risk of developing digital ulcers in a population of 200 Italian systemic sclerosis patients and was built using the Multifactor Dimensionality Reduction (MDR) kernel [10]. The model concerns two SNPs (IL-2 C-330G SNP and IL-6 G-174C SNP, hereafter referred to as IL2 and IL6), and one non-SNP mutation (HLA-B35, hereafter referred to B35). Due to the complexity of B35, only the presence or absence of a particular mutant allele (HLA-B*3501) is recorded; we refer to the absence of this allele as *AA*, and its presence as *Aa*/*aa*.

In each cell of Figure 6, there are two bars. The left bars indicate the frequency of patients (cases) with digital ulcers, and the right bars indicate the frequency of patients without digital ulcers (the controls). If the ratio of cases to controls exceeds a certain threshold, patients are labeled as high risk (which are the dark-shaded cells), otherwise they are low risk (the light-shaded cells).

We want to use our genetic algorithm to learn a Petri net model corresponding to the architecture in Figure 3 that shows how the various genotypes could lead to either a high risk or low risk of the disease, for each of the 18 genotypes in the matrix.

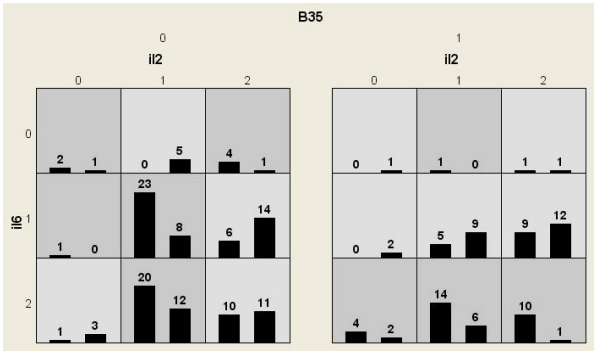


**Fig. 6.** Multifactor dimensionality reduction (MDR) model of non-linear gene-gene interaction. Key: For IL2 and IL6, cell indices 0, 1 and 2 denote genotypes *AA*, *Aa* and *aa* respectively. For B35, cell index 1 indicates *Aa*/*aa* and index 0 indicates genotype *AA*.

### 3.2 Results and Analysis

We performed 32 runs of our genetic algorithm. The maximum fitness value after a run obtained was 0.85, and the minimum was 0.64. The mean best fitness value was 0.70. We examined the Petri net, depicted in Figure 7, with the maximum fitness of 0.85. This net required 5,558 generations to learn, and it has 40 arcs. Rather than showing the net graphically, which would be difficult to interpret, we present it instead as a list of transitions.

$T_{B35}$: $AP_{B35}(1) \Rightarrow PP_{B35}(g_{B35})$
$T_{IL6}$: $AP_{IL6}(1) \Rightarrow PP_{IL6}(g_{IL6})$
$T_{IL2}$: $AP_{IL2}(1), P_1(i) \Rightarrow PP_{IL2}(g_{IL2})$
$T_3$: $PP_{B35}(8), PP_{IL6}(9) \Rightarrow P_1(10)$
$T_4$: $PP_{IL2}(7) \Rightarrow P_1(3), AP_{IL2}(9)$
$T_5$: $PP_{IL6}(3) \Rightarrow P_1(4), AP_{B35}(4), AP_{IL2}(8), AP_{IL6}(9)$
$T_6$: $PP_{IL2}(7), PP_{IL6}(i) \Rightarrow P_1(4)$
$T_7$: $PP_{IL6}(6) \Rightarrow AP_{B35}(3), AP_{IL2}(2)$
$T_8$: $PP_{IL2}(7) \Rightarrow P_1(2), AP_{IL2}(7)$
$T_9$: $P_0(4) \Rightarrow AP_{B35}(9), AP_{IL6}(1)$
$T_{10}$: $PP_{IL2}(3) \Rightarrow P_1(3)\ AP_{IL6}(7)$
$T_{11}$: $PP_{IL2}(7) \Rightarrow P_1(2), AP_{IL6}(2)$
$T_{12}$: $PP_{IL2}(3), PP_{IL6}(i) \Rightarrow P_1(6), AP_{IL2}(10)$

**Fig 7.** Best Petri net obtained with our genetic algorithm. In the figure, $P_0$ is the initial token source for the net, and $P_1$ is the toxic output that is thresholded to determine the risk assessment. The LHS of each transition is a list of inputs and weights (or *i* if the input is an inhibitor), and the RHS is a list of outputs and weights. $AP_k$ and $PP_k$ are the activating and product places of gene *k*, and $g_k$ denotes the genotype-controlled weight for gene *k*.

We generated all 18 state trees (each state tree being derived from one genotype, as described in Section 2), and calculated the number of different outcomes (leaf nodes) for each tree. Of those, the number that gave the correct risk assessment (high or low

**Table 1.** Analysis by genotype of each state tree derived from the Petri net depicted in Figure 7.

| B35 | IL2 | IL6 | Correct | Total | % |
|---|---|---|---|---|---|
| *AA* | *AA* | *AA* | 4 | 4 | 100.0 |
| *AA* | *AA* | *Aa* | 50 | 54 | 92.6 |
| *AA* | *AA* | *aa* | 36 | 39 | 92.3 |
| *AA* | *Aa* | *AA* | 4 | 4 | 100.0 |
| *AA* | *Aa* | *Aa* | 95 | 99 | 96.0 |
| *AA* | *Aa* | *aa* | 57 | 61 | 93.4 |
| *AA* | *aa* | *AA* | 4 | 4 | 100.0 |
| *AA* | *aa* | *Aa* | 68 | 68 | 100.0 |
| *AA* | *aa* | *aa* | 56 | 60 | 93.3 |
| *Aa/aa* | *AA* | *AA* | 2 | 2 | 100.0 |
| *Aa/aa* | *AA* | *Aa* | 2 | 2 | 100.0 |
| *Aa/aa* | *AA* | *aa* | 0 | 2 | 0.0 |
| *Aa/aa* | *Aa* | *AA* | 2 | 2 | 100.0 |
| *Aa/aa* | *Aa* | *Aa* | 2 | 2 | 100.0 |
| *Aa/aa* | *Aa* | *aa* | 2 | 2 | 100.0 |
| *Aa/aa* | *aa* | *AA* | 2 | 2 | 100.0 |
| *Aa/aa* | *aa* | *Aa* | 2 | 2 | 100.0 |
| *Aa/aa* | *aa* | *aa* | 2 | 2 | 100.0 |

according to the model in Figure 6) were computed for each genotype. This enabled us to compute an overall average accuracy across all of the genotypes of 92.6% for this net. The results of the genotype-by-genotype analysis are given in Table 1. As can be observed, the net performs well for all but one of the genotypes.

Finally, we also examined the significance of each individual transition in the network. Taking the Petri net depicted in Figure 7, and iteratively deleting each transition and all arcs incident on it achieved this. We then recomputed the value of the network without the transition, before replacing the transition and its arcs. The recomputed values give an indication of each transition's significance, and can be compared to the original value of 0.85. Table 2 lists, for each transition, these recomputed values. The lower the value of the transition, the greater its significance on the Petri net's dynamics.

As can be observed in Table 2, transitions $T_{B35}$, $T_{IL6}$ and $T_9$ are the most significant: remove them, and the network fails almost completely. This is clearly because the source of the initial tokens ($P_0$) is only accessible via $T_9$, which feeds into B35 and IL6 gene units. All of the remaining transitions except for the final three have different degrees of significance. Interestingly, the final three transitions, $T_4$, $T_8$, and $T_{11}$, have almost no significant impact on the Petri net's behaviour. They could, therefore, be entirely removed, thereby reducing the size of the net from 40 arcs to 31 arcs. This transition analysis could be employed during the genetic algorithm search process itself in order to reduce the size of Petri nets without relying on the random mutation operator.

**Table 2.** Recomputed values of the Petri Net by transition.

| Transition | Value |
|---|---|
| $T_{B35}$ | 0.25 |
| $T_{IL6}$ | 0.25 |
| $T_9$ | 0.25 |
| $T_{IL2}$ | 0.42 |
| $T_7$ | 0.56 |
| $T_3$ | 0.58 |
| $T_6$ | 0.66 |
| $T_{10}$ | 0.67 |
| $T_5$ | 0.67 |
| $T_{12}$ | 0.76 |
| $T_4$ | 0.84 |
| $T_8$ | 0.84 |
| $T_{11}$ | 0.84 |

## 4. Conclusion

We have shown how to construct a Petri net-based model of a set of concurrent processes. Unlike previous approaches to Petri net learning that require global

coordination (in the form of transition priorities or randomised transition firing), our approach "unrolls" the non-determinism by converting a Petri net into a set of state trees, and evaluates the trees rather than the net. This gives an indication of the net's behavior when the transitions are allowed to fire concurrently or in any order whatsoever – a situation most suitable for modeling real world processes.

We have applied this approach to the modeling of non-linear gene interactions, and shown that not only is this approach computationally feasible for a practical application, but also that the analysis of the Petri net's learned using this method may lead to useful insight into the problem being modeled.

## References


1. Beretta L, Santaniello A, Mayo M, Cappiello F, Marchini M and Scorza R (2009.) Genetic and Biological Models of Epistasis to Predict Digital Ulcer Occurrence in Italian Systemic Sclerosis Patients. Article In Submission to Annals of Human Genetics.
2. Cheng S, Yeh H, Lin Y, Lin S, Soo V. (2007). Inferring Gene Regulatory Networks from Microarray Data Based on Transcription Factor Analysis and Conditional Independency. BIOCOMP 2007: 65-71
3. Fishman D, Faulds G, Jeffery R, et al. (1998) The effect of novel polymorphisms in the interleukin-6 (IL-6) gene on IL-6 transcription and plasma IL-6 levels, and an association with systemic-onset juvenile chronic arthritis. J Clin Invest. 102:1369-76.
4. Goldberg, D (1989.) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.
5. Hoffmann SC, Stanley EM, Darrin Cox E, et al. (2001) Association of cytokine polymorphic inheritance and in vitro cytokine production in anti-CD3/CD28-stimulated peripheral blood lymphocytes. Transplantation. 72:1444-50..
6. Lin Y, Yeh H, Cheng S, and Soo V. (2007). Comparing Cancer and Normal Gene Regulatory Networks Based on Microarray Data and Transcription Factor Analysis. In Proc. of the 7th IEEE International Conference on Bioinformatics and Bioengineering, BIBE 2007, pp. 151-157.
7. Mayo M. (2005). Learning Petri net models of non-linear gene interactions. BioSystems, 82(1), 74-82.
8. McGarry K, Loutfi M and Moscardini A. (2007). Stochastic Simulation of the Regulatory Pathways involved in Diabetes using Petri-nets. In Proc. of the International Conference on Computer Theory and Applications (ICCTA2007), Alexandria, Egypt.
9. Moore J, and Hahn L. (2003.) Petri net modelling of high-order genetic systems using grammatical evolution. BioSystems 72, 177–186
10. Moore J. (2004.) Computational analysis of gene-gene interactions using multifactor dimensionality reduction. Expert Review of Molecular Diagnostics 4:6, pp. 795-803.
11. Phillips, PC. (2008) Epistasis--the essential role of gene interactions in the structure and evolution of genetic systems. Nat Rev Genet. 9:855-67.
12. Pociot F, Molvig J, Wogensen L, Worsaae H, Nerup J (1995). A TaqI polymorphism in the human interleukin-1β (IL-1β) gene correlates with IL-1β secretion in vitro. Eur J Clin Invest. 22:396–402.
13. Reisig W. (1985.) Petri nets: an introduction. In: EATCS Monographs on Theoretical Computer Science. Springer-Verlag.
14. Mayo M. and Beretta L. (2009.) Modelling Epistasis in Genetic Disease using Petri Nets, Evolutionary Computation and Frequent Itemset Mining. In submission, Expert Systems with Applications: An International Journal.