Systems Biology Modeling in Human Genetics Using Petri Nets and Grammatical Evolution

Jason H. Moore and Lance W. Hahn

Center for Human Genetics Research, Department of Molecular Physiology and Biophysics, 519 Light Hall, Vanderbilt University, Nashville, TN, USA 37232-0700
{Moore, Hahn}@chgr.mc.Vanderbilt.edu

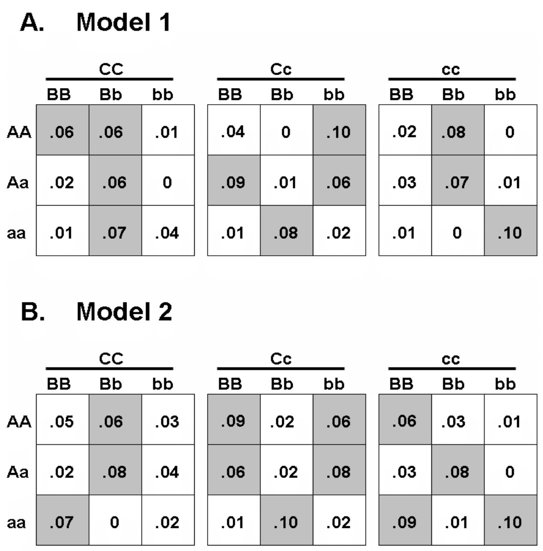
Abstract. Understanding the hierarchical relationships among biochemical, metabolic, and physiological systems in the mapping between genotype and phenotype is expected to improve the diagnosis, prevention, and treatment of common, complex human diseases. We previously developed a systems biology approach based on Petri nets for carrying out thought experiments for the generation of hypotheses about biological network models that are consistent with genetic models of disease susceptibility. Our systems biology strategy uses grammatical evolution for symbolic manipulation and optimization of Petri net models. We previously demonstrated that this approach routinely identifies biological systems models that are consistent with a variety of complex genetic models in which disease susceptibility is determined by nonlinear interactions between two DNA sequence variations. However, the modeling strategy was generally not successful when extended to modeling nonlinear interactions between three DNA sequence variations. In the present study, we develop a new grammar that uniformly generates Petri net models across the entire search space. The results indicate that choice of grammar plays an important role in the success of grammatical evolution searches in this bioinformatics modeling domain.

1 Introduction

Understanding how interindividual differences in DNA sequences map onto interindividual differences in phenotypes is a central focus of human genetics. Genotypes contribute to the expression of phenotypes through a hierarchical network of biochemical, metabolic, and physiological systems. The availability of biological information at all levels in the hierarchical mapping between genotype and phenotype has given rise to a new field called systems biology. One goal of systems biology is to develop a bioinformatics framework for integrating multiple levels of biological information through the development of theory and tools that can be used for mathematical modeling and simulation [1]. The promise of both human genetics and systems biology is improved human health through the improvement of disease diagnosis, prevention, and treatment.

Central to any study of common human disease is the realization that the mapping relationship between genotype and phenotype is extremely complex. Part of this complexity is due to epistasis or nonlinear gene-gene interactions. The historical biological definition of epistasis is one gene masking or standing upon the effects of another gene [2] while the statistical definition is a deviation from additivity in a linear model [3]. Today, epistasis is believed to be a ubiquitous component of the genetic architecture of common human diseases [4]. As a result, the identification of genes with genotypes that confer an increased susceptibility to a common disease will require a research strategy that embraces, rather than ignores, the complexity of these diseases.

We have developed a computational systems biology strategy for carrying out thought experiments about the complexity of biological systems that are consistent with a given genetic model [5-8]. With this approach, discrete dynamic systems modeling is implemented using Petri nets. Symbolic manipulation and optimization of Petri net models is carried out using grammatical evolution. This approach routinely generated Petri net models that were consistent with a variety of genetic models in which disease susceptibility is dependent on nonlinear interactions between two DNA sequence variations [5,6]. However, when applied to higher-order genetic models, the strategy did not consistently yield perfect Petri nets [7]. In fact, only one perfect model was discovered out of 100 independent runs. The goal of the present study is to develop a strategy that is able to discover Petri net models of biological systems that are consistent with nonlinear interactions between three DNA sequence variations.



A. Model 1

	CC			Cc			cc		
	BB	Bb	bb	BB	Bb	bb	BB	Bb	bb
AA	.06	.06	.01	.04	0	.10	.02	.08	0
Aa	.02	.06	0	.09	.01	.06	.03	.07	.01
aa	.01	.07	.04	.01	.08	.02	.01	0	.10

B. Model 2

	CC			Cc			cc		
	BB	Bb	bb	BB	Bb	bb	BB	Bb	bb
AA	.05	.06	.03	.09	.02	.06	.06	.03	.01
Aa	.02	.08	.04	.06	.02	.08	.03	.08	0
aa	.07	0	.02	.01	.10	.02	.09	.01	.10

Fig. 1. Penetrance functions for nonlinear gene-gene interaction models 1 (A) and 2 (B). High-risk genotype combinations are shaded while low-risk combinations are unshaded.

2 The Nonlinear Gene-Gene Interaction Models

Our two high-order, nonlinear, gene-gene interaction models are based on penetrance functions. Penetrance functions represent one approach to modeling the relationship between genetic variations and risk of disease. Penetrance is simply the probability (P) of disease (D) given a particular combination of genotypes (G) that was inherited (i.e. P[D|G]). A single genotype is determined by one allele (i.e. a specific DNA sequence state) inherited from the mother and one allele inherited from the father. For most genetic variations, only two alleles (encoded by *A* or *a*) exist in the biological population. Therefore, because the ordering of the alleles is unimportant, a genotype can have one of three values: *AA*, *Aa* or *aa*. Figures 1A and 1B (above) illustrate the penetrance functions used for Models 1 and 2, respectively. Each model was discovered using a modified version of the software of Moore et al. [9]. What makes these models interesting is that disease risk is dependent on each particular combination of three genotypes. Each single-locus genotype has effectively no main effect on disease risk when the allele frequencies are equal and the genotypes are consistent with Hardy-Weinberg proportions.

3 An Introduction to Petri Nets for Modeling Discrete Dynamic Systems

Petri nets are a type of directed graph that can be used to model discrete dynamical systems [10]. Goss and Peccoud [11] demonstrated that Petri nets could be used to model molecular interactions in biochemical systems. The core Petri net consists of two different types of nodes: places and transitions. Using the biochemical systems analogy of Goss and Peccoud [11], places represent molecular species. Each place has a certain number of tokens that represent the number of molecules for that particular molecular specie. A transition is analogous to a molecular or chemical reaction and is said to fire when it acquires tokens from a source place and, after a possible delay, deposits tokens in a destination place. Tokens travel from a place to a transition or from a transition to a place via arcs with specific weights or bandwidths. While the number of tokens transferred from place to transition to place is determined by the arc weights (or bandwidths), the rate at which the tokens are transferred is determined by the delay associated with the transition. Transition behavior is also constrained by the weights of the source and destination arcs. A transition will only fire if two preconditions are met: 1) if the source place can completely supply the capacity of the source arc and, 2) if the destination place has the capacity available to store the number of tokens provided by the full weight of the destination arc. Transitions without an input arc, act as if they are connected to a limitless supply of tokens. Similarly, transitions without an output arc can consume a limitless supply of tokens. The firing rate of the transition can be immediate, delayed deterministically or delayed stochastically, depending on the complexity needed. The fundamental behavior of a Petri net can be controlled by varying the maximum number of tokens a place can hold, the weight of each arc, and the firing rates of the transitions.

4 Our Petri Net Modeling Strategy

The goal of identifying Petri net models of biochemical systems that are consistent with observed population-level gene-gene interactions is accomplished by developing Petri nets that are dependent on specific genotypes from two or more genetic variations. Here, we make transition firing rates and/or arc weights genotype-dependent yielding different Petri net behavior. Each Petri net model is related to the genetic model using a discrete version of the threshold model from population genetics [12]. With a classic threshold or liability model, it is the concentration of a biochemical or environmental substance that is related to the risk of disease, under the hypothesis that risk of disease is greatly increased once a particular substance exceeds some threshold concentration. Conversely, the risk of disease may increase in the absence of a particular factor or with any significant deviation from a reference level. In such cases, high or low levels are associated with high risk while an intermediate level is associated with low risk. Here, we use a discrete version of this model for our deterministic Petri nets. For each model, the number of tokens at a particular place is recorded and if they exceed a certain threshold, the appropriate risk assignment is made. If the number of tokens does not exceed the threshold, the alternative risk assignment is made. The high-risk and low-risk assignments made by the discrete threshold from the output of the Petri net can then be compared to the high-risk and low-risk genotypes from the genetic model. A perfect match indicates the Petri net model is consistent with the gene-gene interactions observed in the genetic model. The Petri net then becomes a model that relates the genetic variations to risk of disease through an intermediate biochemical network.

5 The Grammatical Evolution Algorithm

5.1 Overview of Grammatical Evolution

Evolutionary computation arose from early work on evolutionary programming [13,14] and evolution strategies [15,16] that used simulated evolution for artificial intelligence. The focus on representations at the genotypic level lead to the development of genetic algorithms by Holland [17,18] and others. Genetic algorithms have become a popular machine intelligence strategy because they can be effective for implementing parallel searches of rugged fitness landscapes [19]. Briefly, this is accomplished by generating a random population of models or solutions, evaluating their ability to solve the problem at hand, selecting the best models or solutions, and generating variability in these models by exchanging model components between different models. The process of selecting models and introducing variability is iterated until an optimal model is identified or some termination criteria are satisfied. Koza [20] developed an alternative parameterization of genetic algorithms called genetic programming where the models or solutions are represented by binary expression trees. Koza [21] and others [22] have applied genetic programming to modeling metabolic networks.

Grammatical evolution (GE) has been described by O'Neill and Ryan [23, 24] as a variation on genetic programming. Here, a Backus-Naur Form (BNF) grammar is specified that allows a computer program or model to be constructed by a simple genetic algorithm operating on an array of bits. The GE approach is appealing because only a text file specifying the grammar needs to be altered for different applications. There is no need to modify and recompile source code during development once the fitness function is specified.

5.2 A Grammar for Petri Net Models in Backus-Naur Form

Moore and Hahn [5-8] developed a grammar for Petri nets in BNF. Backus-Naur Form is a formal notation for describing the syntax of a context-free grammar as a set of production rules that consist of terminals and nonterminals [25]. Nonterminals form the left-hand side of production rules while both terminals and nonterminals can form the right-hand side. A terminal is essentially a model element while a nonterminal is the name of a possibly recursive production rule. For the Petri net models, the terminal set includes, for example, the basic building blocks of a Petri net: places, arcs, and transitions. The nonterminal set includes the names of production rules that construct the Petri net. For example, a nonterminal might name a production rule for determining whether an arc has weights that are fixed or genotype-dependent. We show below the production rule that was executed to begin the model building process for the study by Moore and Hahn [7] and then describe the modifications evaluated in the present study.

```
<root> ::= <pick_a_gene> <pick_a_gene> <pick_a_gene> <net_iterations>
<expr> <transition> <transition> <place_noarc>
```

When the initial <root> production rule is executed, a single Petri net place with no entering or exiting arc (i.e. <place_noarc>) is selected and a transition leading into or out of that place is selected. The arc connecting the transition and place can be dependent on the genotypes of the genes selected by <pick_a_gene>. The nonterminal <expr> is a function that allows the Petri net to grow. The production rule for <expr> is shown below.

```
<expr> ::= <expr> <expr>      0
         | <arc>              1
         | <transition>       2
         | <place>            3
```

Here, the selection of one of the four nonterminals (0, 1, 2, or 3) on the right-hand side of the production rule is determined by a combination of bits in the genetic algorithm chromosome.

The base or minimum Petri net that is constructed using the <root> production rule consists of a single place, two transitions, and an arc that connects each transition to the place. Multiple calls to the production rule <expr> by the genetic algorithm chromosome can build any connected Petri net. In addition, the number of times the Petri net is to be iterated is selected with the nonterminal <net_iterations>. Many other production rules define the arc weights, the genotype-dependent arcs and transitions, the number of initial tokens in a place, the place capacity, etc. All decisions made in the building of the Petri net model are made by each subsequent bit or combination of bits in the genetic algorithm chromosome. The complete grammar is too large for detailed presentation here but can be obtained from the authors upon request.

While the grammar described above was successful for modeling interactions among two DNA sequence variations [5, 6], it was not successful for modeling interactions among three DNA sequence variations [7]. A potential concern is that the grammar starts with very a simple Petri net in the root production rule. Through the use of <expr>, the grammar is theoretically capable of generating larger, more complex Petri nets. However, there is clearly a bias towards smaller, simpler Petri nets. For example, Moore and Hahn [6] discovered Petri nets that almost always consisted of one place, two transitions, and two arcs. In the present study, we have modified the grammar to uniformly generate Petri nets from a defined search space. The new grammar builds a Petri net with one to five places, one to 20 transitions, and four to 24 arcs. Each architecture within these limits is equally probable. We also require that at least one conditional element be present for each of the DNA sequence variations in the genetic model.

5.3 The Fitness Function

Once a Petri net model is constructed using the BNF grammar, as instructed by the genetic algorithm chromosome, the model fitness is determined. As described in detail by Moore and Hahn [6, 7], this is carried out by executing the Petri net model for each combination of genotypes in the genetic dataset and comparing the final token counts at a defined place to a threshold constant to determine the risk assignments. The optimal threshold is determined by systematically evaluating different threshold constants. Fitness of the Petri net model is determined by comparing the high risk and low risk assignments made by the Petri net to those from the given nonlinear gene-gene interaction model (e.g. see Figure 1). Fewer inconsistencies are associated with a better fitness. Ideally, the risk assignments made by the Petri net are perfectly consistent with those in the genetic model.

5.4 The Genetic Algorithm Parameters

Details of the genetic algorithm are given by Moore and Hahn [7]. Briefly, each genetic algorithm chromosome consisted of 14 32-bit bytes. In implementing the grammar, it is possible to reach the end of a chromosome with an incomplete instance of the grammar. To complete the instance, chromosome wrap-around was used [23, 24].

In other words, the instance of the grammar was completed by reusing the chromosome as many times as was necessary.

We ran the genetic algorithm a total of 100 times with different random seeds for each gene-gene interaction model. Each run consisted of a maximum of 800 generations. The genetic algorithm was stopped when a model with a classification error of zero was discovered. We used a parallel search strategy of 10 demes, each with a population size of 2000, for a total population size of 20,000. A best chromosome migrated from each deme to all other demes every 25 generations. The recombination probability was 0.6 while the mutation probability was 0.02.

6 Results

The grammatical evolution algorithm was run a total of 100 times for each of the two high-order, nonlinear gene-gene interaction models. For genetic model one, the modified grammatical evolution strategy yielded 32 perfect Petri net models. For genetic model two, 36 perfect Petri net models were discovered. These results are in contrast to the grammar of Moore and Hahn [7] that yielded no perfect Petri nets for the first genetic model and only one perfect Petri net for the second genetic model.

Table 1 below summarizes the mode (i.e. most common) and range of the number of places, arcs, transitions, and conditionals that define the genotype-dependencies of the elements in the best Petri net models found across the 100 runs for each model. For each gene-gene interaction model, most Petri net models discovered consisted of one place and a minimum of 11 transitions, seven arcs, and seven elements that are conditional on genotype. In general, these Petri net models were much larger than those described by Moore and Hahn [7] due to the improved grammar that samples the search space in a uniform manner. Figure 2 illustrates example Petri net architectures that were discovered by the grammatical evolution algorithm for each genetic model. All generated Petri net models are available upon request.

Table 1. Summary of the distribution (mode and range) of the number of different Petri net elements identified across 100 grammatical evolution runs for the two nonlinear gene-gene interaction models.

	Mode (range) number of Petri net elements	
Petri net element	Model 1	Model 2
Place	1 (1-5)	1 (1-4)
Arc	7 (5-30)	12 (5-24)
Transition	19 (3-25)	11 (3-22)
Conditional	7,10 (5-29)	10 (5-30)

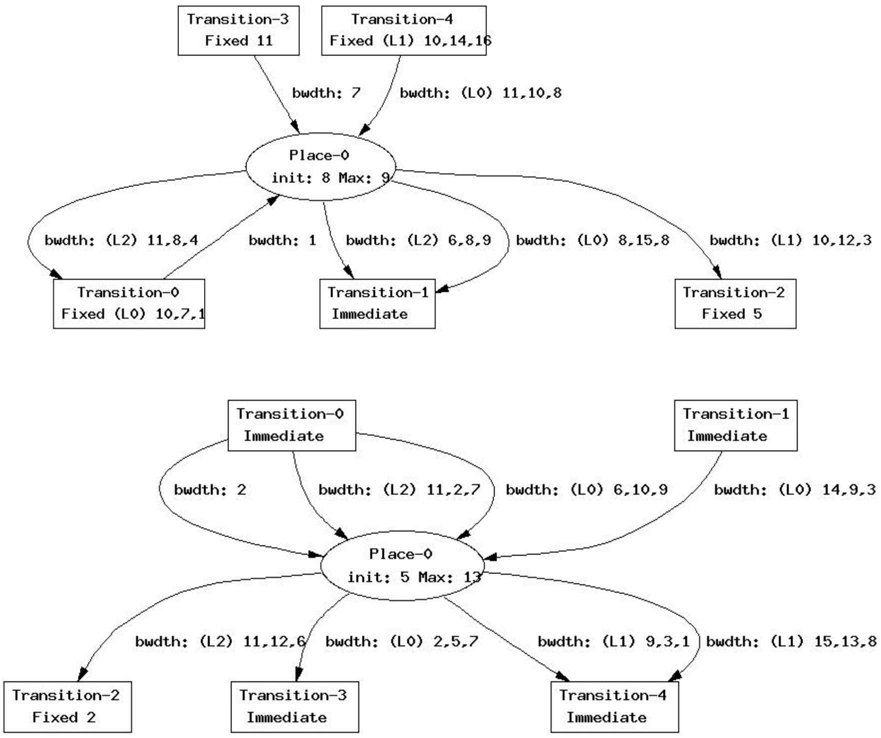


Fig. 2. Examples of Petri nets discovered for genetic model 1 (top) and 2 (bottom). Each place has an initial (init) and maximum (max) token count. Transitions have either a fixed firing rate of one token every one (immediate) to 16 time steps or a firing rate that is dependent on the genotype (e.g. 10, 14, 16 for *AA, Aa, aa*, respectively) at a particular DNA sequence variation (e.g. L0, L1, or L2). Similarly, arcs have a bandwidth (bwdth) that is either fixed (e.g. 7) or is dependent on the genotype (e.g. 11, 10, 8 for *AA, Aa, aa*, respectively) at a particular DNA sequence variation.

7 Discussion

The main conclusion of this study is that a grammar that generates Petri net models in a uniform fashion across the search space of all possible models significantly outperforms a grammar that relies on <expr> in the production rule to build Petri net complexity from an initial simple starting point. This study points to the importance of grammar selection on the ultimate performance of the search. In fact, choice of grammar may sometimes be more important than parameters such as the number of generations used in the evolutionary computing search. For example, the same parameter settings (described in Section 5.4 above) were used in the present study and the study by Moore and Hahn [7]. The only difference was the how the grammar was

written. The present study yielded perfect models more than 30% of the time compared
to nearly 0-1% in the previous study. To illustrate this point, we reran the previous study [7] using twice the number of generations (1600 instead of 800) and found only a very small improvement with a new success rate of 1-5%. Also, using a longer GA chromosome to avoid wraparound and fixed length codons as recommended [24] did not significantly improved the success rate. It is certainly advantageous to optimize the grammar rather than the search parameters since the latter usually results in greatly increased computational time.

What is the next step for this modeling approach? Although the Petri net strategy identified perfect models, it did not do so routinely. The next step is to further optimize the grammatical evolution search in order to identify perfect models on every run. This will be necessary if this approach is to be extended to even higher-order genetic models. Changes to the grammar in addition to changes to the search parameters will be explored and evaluated.

The results of this study will play an important role in the development of complex biological systems modeling tools that can be used to carry out thought experiments about genotype to phenotype relationships. Thought experiments can play an important role in scientific discovery by generating testable hypotheses [26]. Once a biological hypothesis is formulated, it can then be evaluated using perturbation experiments in model organisms [27]. We anticipate both Petri nets and evolutionary computing approaches such as grammatical evolution will be useful for systems biology modeling in the domain of human genetics.

Acknowledgements. This work was supported by National Institutes of Health grants HL65234, HL65962, GM31304, AG19085, and AG20135.

References

1. Ideker, T., Galitski, T., Hood, L.: A new approach to decoding life: systems biology. Annual Review of Genomics and Human Genetics 2 (2001) 343-72
2. Bateson, W.: Mendel's Principles of Heredity. Cambridge University Press, Cambridge (1909)
3. Fisher, R.A.: The correlation between relatives on the supposition of Mendelian inheritance. Transactions of the Royal Society of Edinburgh 52 (1918) 399-433
4. Moore, J.H. The ubiquitous nature of epistasis in determining susceptibility to common human diseases. Human Heredity 56 (2003) 73-82
5. Moore, J.H., Hahn, L.W. Grammatical evolution for the discovery of Petri net models of complex genetic systems. In: Cantu-Paz et al., editors, Genetic and Evolutionary Computation – GECCO 2003. Lecture Notes in Computer Science 2724 (2003) 2412-13
6. Moore, J.H., Hahn, L.W. Evaluation of a discrete dynamic systems approach for modeling the hierarchical relationship between genes, biochemistry, and disease susceptibility. Discrete and Continuous Dynamical Systems: Series B 4 (2004) 275-87
7. Moore, J.H., Hahn, L.W. Petri net modeling of high-order genetic systems using grammatical evolution. BioSystems 72 (2003) 177-86

8. Moore, J.H., Hahn, L.W. An improved grammatical evolution strategy for Petri net modeling of complex genetic systems. In: G.R. Raidl et al., editors, Applications of Evolutionary Computing, Lecture Notes in Computer Science 3005 (2004) 62-71
9. Moore, J.H., Hahn, L.W., Ritchie, M.D., Thornton, T.A., White, B.C: Application of genetic algorithms to the discovery of complex genetic models for simulation studies in human genetics. In: W.B.Langdon et al., editors, Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann Publishers, San Francisco (2002)
10. Desel, J., Juhas, G.: What is a Petri net? Informal answers for the informed reader. In H. Ehrig, G. Juhas, J. Padberg, and G. Rozenberg, editors, Unifying Petri Nets, Lecture Notes in Computer Science 2128, pp 1-27. Springer (2001)
11. Goss, P.J., Peccoud, J.: Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. Proceedings of the National Academy of Sciences USA 95 (1998) 6750-5
12. Falconer, D.S., Mackay, T.F.C: Introduction to Quantitative Genetics, 4th edition, Longman, Essex (1996)
13. Fogel, L.J. Autonomous automata. Industrial Research 4 (1962) 14-19
14. Fogel, L.J., Owens, A.J., Walsh, M.J. Artificial Intelligence through Simulated Evolution. John Wiley, New York (1966)
15. Rechenberg, I. Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Farnborough, U.K., Library Translation No. 1122 (1965)
16. Schwefel, H.-P. Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik. Diploma Thesis, Technical University of Berlin (1965)
17. Holland, J.H. Adaptive plans optimal for payoff-only environments. In: Proceedings of the 2nd Hawaii International Conference on Systems Sciences. University of Hawaii, Honolulu (1969) 917-920
18. Holland, J.H. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
19. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Reading: Addison-Wesley (1989)
20. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge London (1992)
21. Koza, J.R., Mydlowec, W., Lanza, G., Yu, J., Keane, M.A. Reverse engineering of metabolic pathways from observed data using genetic programming. Pacific Symposium on Biocomputing 6 (2001) 434-45
22. Kitagawa, J., Iba, H. Identifying metabolic pathways and gene regulation networks with evolutionary algorithms. In: Evolutionary Computation and Bioinformatics. Fogel, G.B., Corne, D.W., editors. Morgan Kaufmann Publishers, San Francisco (2003) 255-278
23. O'Neill, M., Ryan, C.: Grammatical evolution. IEEE Transactions on Evolutionary Computation 5 (2001) 349-358
24. O'Neill, M., Ryan, C.: Grammatical evolution: Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers, Boston (2003)
25. Marcotty, M., Ledgard, H.: The World of Programming Languages. Springer-Verlag, Berlin (1986)
26. Di Paolo, E.A., Noble, J., Bullock, S.: Simulation models as opaque thought experiments. In: Proceedings of the Seventh International Conference on Artificial Life. Dedau, M.A. et al, editors. The MIT Press, Cambridge (2000)
27. Jansen, R.C.: Studying complex biological systems using multifactorial perturbation. Nature Reviews Genetics 4 (2003) 145-51