Whatever Emerges should be Intrinsically Useful

Hugues Bersini

IRIDIA-CP 194/6
Université Libre de Bruxelles
50, av. Franklin Roosevelt – 1050 Bruxelles
Belgium
bersini@ulb.ac.be

Abstract

The practical work presented in this paper uses a GA to evolve a cellular automata (CA) implementation of a binary numbers adder. One very useful way to compress the enormous search space and eventually find an optimal CA consists in adopting a macro-coding of the states and the rule table. It is further discussed how this work illustrates and defends our favorite position in the currently vivid epistemological debate around the notion of "emergence". This position is Crutchfield's "intrinsic emergence" one, in which to say that a macro-property is emergent requires that this "property" supplies some mechanical and non-human observer with additional functionality.

Introduction and Intrinsic Emergence

For many years now, cellular automata (CA) [16] have been the favorite computational platform to experiment and illustrate emergent phenomena. It is far from surprising that many authors have relied on their CA experimentation to quest for formal definitions of the nature of "emergence" and to practically validate them [3][10][14]. This paper is following a similar trend by fully adopting the practice of CA. On the whole, all authors interested in the rationalization of emergence converge to the fact that at least two levels of observation are required: A first one in which the micro-states and micro behavioral rules are specified and implemented, and a second one, which by only depending upon the underlying micro-characteristics, exhibits interesting macro-phenomena. They are obtained by unfolding in space and time the micro-rules though the micro-states, most of the time in a non-decomposable way (see [10] for an attempt to formalize this non-decomposability).

An observer, so far always human, is necessary to instantiate this second and more abstract level of observation and to spot, follow and trace these interesting and new phenomena. This characterization of emergence has turned out to be quite common [2][3][8][10][13] and could be symbolized by the little UML class diagram of figure 1 showing the three basic actors: the parts, the whole, allowing to iterate the parts in space and time, and the human observer. Nevertheless, this paper considers that such a classical characterization, though including necessary ingredients (i.e. the two levels of observation and the abstraction in space and time of the second with respect to the first), is far from sufficient, severely limited and incomplete on one essential aspect: the identity and the role of this second level observer. The problem is not so much that "the whole is more than the sum of its parts" but rather who is responsible for observing that "whole". For CA, Neural Networks, other computer simulations of networks and whatever computational source of emergence, the observer is generally accepted to be human. However, this "anthropomorphisation" of the phenomenon of emergence is antagonistic to any scientific practice that, in principle, aims at not leaving subjectivism a leg to stand on. Basically, if the formalization of emergence demands the intervention of a human observer, even worse to be "psychologically surprised" [14], its intrusion in the vocabulary of physics is compromised right off the bat.

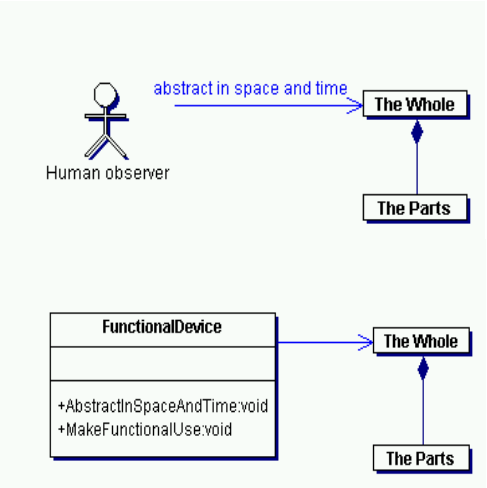

Fig. 1. The "Intrinsic Emergence": from the human observer to the functional device.

To our knowledge, such a limitation has been faced and removed mainly by two authors [5][6][7] who, in their writing, have answered this preoccupation by supplying the characterization of emergence with a key ingredient. Like the updated UML class diagram shows, a "functional device" must substitute the human observer that, for whatever utility or performance reasons, will fine-tune its observation of any macro-phenomena produced by the system. Cariani [5] claims that, for a phenomenon to be said emergent, devices need to be built, able to find new observable, autonomously relative to us, whose selection and tuning must rely on performance measures. However the most convincing reply to this limitation and which provides the main guidelines for the work to be described in this paper is the Crutchfield's definition of "intrinsic emergence" [6]:

"*… Pattern formation is insufficient to capture the essential aspect of the emergence of coordinated behaviour and global information processing… At some basic level though, pattern formation must play a role… What is distinctive about intrinsic emergence is that the patterns formed confer additional functionality which supports global information processing… During intrinsic emergence there is an increase in intrinsic computational capability, which can be capitalized on and so lends additional functionality*."

For instance, the game of life "glider" [12][3][10] should not be characterized as "emergent" unless some functional device able to observe the CA and to aggregate its cells in space and in time (the glider covers 5 cells and is a period-four phenomenon) will make a specific use of it. Emergence is a bottom-up phenomenon but like in any top-down and hierarchical construction of complex systems, some external entity needs to make sense and use of whatever emerges at any level of the construction. Following trends initiated by Packard, Crutchfield, Mitchell and others [1][7][11], the work described in this paper consists in using an evolutionary algorithm to discover a CA able to perform an engineering task, in our case, the binary addition of numbers coded on n bits. It is shown that the discovery of such an efficient CA is very painful by adopting the natural micro-coding of the states and the rules.

An important improvement and acceleration of this discovery is allowed by adopting a simplified macro or abstract characterization of the states and the rules of the CA. This new and one level up "macro observation", intrinsically emerges, since autonomously tuned by the system itself on the basis of performance measures. Once a new macro-way of observing the system is adopted, new phenomena emerge since being detectable only by this new observing device. Allowing part of the evolutionary process to take place by adopting such an emergent observable accelerates in a consequential way the discovery of a quasi-optimal CA. This is why and only why, according to the definition of "intrinsic emergence", this observable can be labeled as "emergent", since it is intrinsically selected by the system to improve its performance and needs no external human observer.

Apart from casting some fresh light on the characterization of emergence, any engineering of distributed computation should often be able to reproduce such a practice: find a way of observing the system which helps the optimizing of its structure and behaviour. The next section will describe the task to be achieved by the CA. The third one will describe how a simple evolutionary algorithm can search for a satisfactory non-uniform CA, finally followed by the two last sections discussing and experimenting how a useful way of observing the CA can emerge in order to boost the discovery of this satisfactory solution.

The CA Task: Binary Addition

Binary addition is an interesting task to be performed by CA for essentially two reasons. First, it is much more reminiscent of what real digital circuits perform in computers than "density classification" or "synchronization" [1][7][11][15], so that the developments presented here could more easily be transposed for the automatic discovery of useable logical circuits. Moreover, the computer digital circuitry for binary additions is built by growing up in functional abstraction i.e. big circuits (able to add big numbers) are composed of small circuits. Basically the little circuits, optimally shaped for adding very small binary numbers, are kept as the basic blocks for building larger circuits. This fact will be important to help at better understanding the type of observables we want our system to be able to "emerge".

The CA used to perform this task is a two dimensional CA (this dimensionality was favored to the one-dimensional case by analogy with digital circuitry and for reasons that will become obvious later on), with periodic boundaries and characterized by the classical 8-cells Moore neighborhood. At time 0, like indicated in fig.2, two n bits binary numbers are installed in the first line of the CA (the first number on the extreme left, the second number on the extreme right, in the figure, numbers are coded on 5 bits). All other cell initial states are tuned to 0. Time step after time step, the CA then updates all its cells in a synchronous way, by complying with the rule table associating with each of the 2^8 possible neighborhood configurations the next value of the state (so that the rule table here has 2^8 entries with value "0" or "1" for each). The goal of the CA global computation is, like indicated in fig.2, after a given number of time steps, taken here to be equal to the vertical size of the CA (the number of lines), to obtain, still in the first line and on its extreme left, the n+1 bits composing the result of the addition of the two numbers.

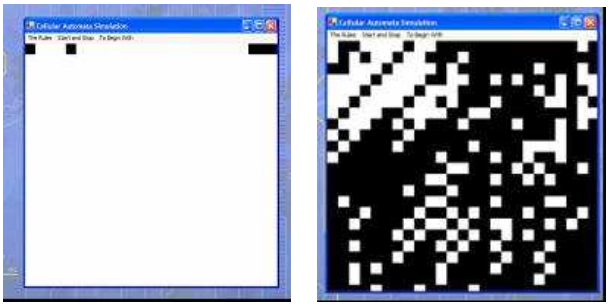

Fig.2. The CA Task: Binary addition of two five bits binary numbers. Initially, the numbers are shown at the extreme left and extreme right of the first line of the first CA (a 25x25 periodic CA). The two numbers are 10001 and 00111. All the other cells are set to 0. After 25 time steps, the correct sum: 011000 should appear at the extreme left of the first line of the second CA.

In the software, all parameters, the size of the CA, the size of the numbers to add, the number of time steps to reach the result, can easily be changed. However, here for sake of clarity, we will maintain one set of number, 25 for the size of a square CA (so that 25 five time steps are computed in order to get the answer) and 5 for the size of the two numbers to add by the same CA. Finally, this addition must be performed for a certain number of couples of binary numbers, say 5 again, like, for instance given below:

```
 N1    N2    N1+N2
01110 11110 101100
00111 10101 011100
11100 00110 100010
11001 01111 101000
10111 11110 110101
```

This way of doing makes CA very similar to classical addition logical circuits were the numbers to add are injected as input on the top of the circuit, then cascade down through a sequence of logical gates to finally give the result of the addition at the bottom of this circuit. It must be clear that this procedure does not guarantee at all obtaining a universal adder but just a specific one, which works well for these n numbers. However, the bigger this n the more likely you'll make the adder universal.

What Kind of GA Evolves the CA

The kind of GA and how well it allows optimizing the rule table is not a central topic here and an infinite number of possible evolutionary algorithms could be applied. The rule table to be optimized is composed of 2^8 bits so that the search must take place in this huge binary space (2^{2^8} possibilities). A population of 20 individuals is being evolved. After computing the fitness of every individual, the best one is kept in the subsequent population (the "elitist version") and the best half of the population is selected on which to apply the mutation and the crossover mechanisms responsible for generating the subsequent population. Following a series of mutation and crossover upon the 10 best individuals, a new population of 20 individuals is generated. More precisely, the first individual of the new population is the same as the first individual of the previous one. Then the second and the third individuals of the new population are obtained by randomly mutating one bit of two individuals arbitrarily taken among the 10 best individuals of the previous population. Since a large preference is given to the crossover mechanism, the 17 remaining individuals are obtained by applying the simplex crossover on three individuals.

The simplex crossover first introduced in [4], and which has been shown to improve on the two parents classical one, is described below.

Simplex Crossover:

1) Take three parents and rank then by decreasing fitness: P1>P2>P3
2) For each bit, if parent 1 and parent 2 agree on the value of the bit, the offspring will have the same value for this bit. If parent 1 and parent 2 disagree on the value of the bit, take the reverse of the value of this bit given by parent 3.
3) The idea is that both parent 1 and 2, the good ones, attempt at shaping the offspring similar to them while parent 3, the worst of all three, and as soon as it can say something, will force the offspring to be very different from him.

For instance:

```
P1:   1001100110
P2:   1011110100
P3:   0001101111
      ----------
Offspring:  1011110100
```

In our case and provided the individuals in the population are ranked by decreasing fitness, the three parents are selected so that the first resides in the first third of the 10 best, the second in the second third and the third in the last third. Again, we are not so much interesting here in how well this GA performs. This is one very simple instance to implement and much better versions could be imagined. It largely suffices to test and validate the main idea behind this work. The fitness is computed by summing, over all five couple of numbers, the Hamming distance between the desired result (the correct sum) and the one obtained after 25 time steps of the CA run. For all possible rule tables and for 5 couples of 5 bit numbers, the fitness value is comprised between 0, the best fitness, and 30, the worst one (i.e. 5*6).

After many attempts, the performances obtained were very poor, giving an average fitness around 6.5. In substance, after thousands of iterations, no CA could be found able to sum these 5 binary numbers. Like done by other authors before [15], a first recovery decision consisted in allowing the CA to become non-uniform i.e. to allow distinct rule tables to characterize the update mechanism of different cells. Then a cell could be characterized by one rule table and its neighboring cell by a distinct one. While this increase in the degrees of freedom should naturally improve the computational capacity of the CA, the original motivation was to be able to replicate the heterogeneity of computer logical circuits (composed of "AND", "NOR", "XOR", etc. gates). Consequently, an additional attribute has to be associated with every cell i.e. which one of the rule tables it complies with. The way the rule tables were distributed among the cells was random and the figure below shows the increase in performance obtained by allowing the number of distinct rule tables to vary from 1 to 5 (the parameter "v" for "variety"). But, still no CA could be found with fitness better than 6 after 500 iterations of the GA search.

The new individual to optimize would now comprise $v*2^8$ bits increasing by many order of magnitude the size of the search space. Indeed, the same figure shows that, although a small increase of v allows an improvement in the fitness, in order to obtain this improvement and to discover better individuals, the number of GA iterations must equally increase in a drastic way (500 GA iterations in the first case and 2000 in the second one). So, while increasing the heterogeneity increases in proportion the chance to find a better CA (for instance with v=5, the average best fitness obtained after 2000 GA iterations is 5,4), the computational time must also be consequently increased to allow this discovery. A very practical problem turns out to be how to accelerate the discovery of an optimal CA in a fundamental way.

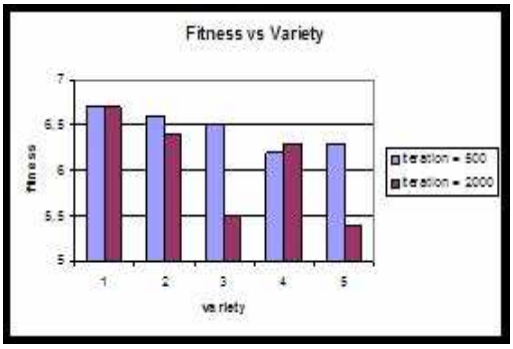

Fig.3. The fitness as a function of the variety of the rule tables for 500 and 2000 GA iterations

Discovery of a Useful Way to Observe CA

Facing this huge search space, among the many ways to reduce its size, the original one proposed in this paper requires to modify the way CA is observed and to move one level up in abstraction to look at the states and the rules. This new observable makes any cell to be characterized by only x out of the 8 states composing its neighborhood and to mask the (8-x) others. For instance, x could be taken to be 3, like indicated in fig.4. If only 3 neighbors are taken into account, many cells, whatever distinct precise neighborhood they have, turn out to be in identical state. This is very reminiscent of the "don't care" symbol of Holland's definition of GA schema. Instead of "10010000", for instance, the state of a cell will become "#0####00". A formidable collapse in the dimension of the search space follows from this abstract observation which, in presence of 5 don't care (we will keep with this value in the following although a bigger or smaller one is equally possible), boils the coding of the rule table down to 2^3*v (v still being the variety factor).

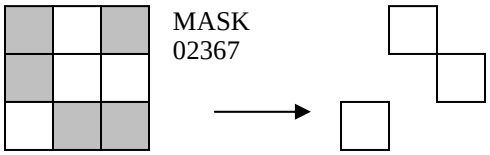

Fig. 4. Here the mask "02367" has been applied

The way a cell is updated now complies with this reduced coding of the rule table and 2^5 distinct neighborhood (the masked "don't care cells") will give the same next state. The (8-x) hidden neighboring cells will be called the "mask" and C_8^5 masks are possible. A mask will be defined, for instance, as "02367" i.e. the five hidden neighboring cells (this is the case shown in fig.4). The first experiments we tried consisted in seeing whether such a reduced search space, although skipping the fine tuning probably necessary at the discovery of an optimal CA, degraded or not the performance obtained so far. The GA was exactly the same apart from tackling a consequently smaller binary space and the 5-cells masks were randomly generated. The experimental results were quite surprising and supported the initiative since, despite this degradation in characterizing the cell state, the reduction of the search space would largely compensate for it. In average, the fitness, following a same number of GA iterations, was slightly improved. For instance, for the "variety" equal to 5, one point of fitness could be gained, around 5 instead of 6.

Emergence of a Useful way to Observe CA

At this stage of the work, and although this masking sounds as a promising proposal, several remarks can be made. First, though slightly improved, the fitness is still unsatisfactory and an average of five is the best we can reach. Second, it is clear that by limiting the observation to such a small part of the cell state, it's hard to imagine

how we could reach one very best individual the coding of which should demand for a fine tuning. Last but not least, with respect to the problematic of emergence, this masking can't be said to emerge since it is manually imposed by the human user.

One way to answer all these remarks is by the following new set of experiments that we indeed performed. First, let's define the emergent "observable" to be the "mask" given the best fitness after a certain number of random trials. In such a way, this masking will be intrinsically selected by the system itself with no need for human intervention. So there should be an initial set of trials to reveal this promising mask. The right mask will emerge out of this initial sequence of random trials. Afterwards, both the mask and the population of the best rules obtained with this mask should be memorized so as to release a new set of simulations in which the complete coding of the cell states and the rules will be re-established. The new algorithm works as follows:

1) For "s" trials, generate one random mask and compute the best fitness after "y" GA iterations
2) Out of these s trials, memorize the best mask and the best rules set for this mask
3) Re-establish the complete coding and generate the initial rule set from the rule set memorized after the "masking" phase. To do so, the $v*2^8$ bits of the "unmask" version bit will be obtained by taking the value of the corresponding $v*2^3$ bits of the memorized "masked" version, so that $v*2^5$ bits will share the same initial value. It is easy to understand that this procedure guarantees that the best fitness of this new unmasked rules set is equal to the best fitness of the previous masked one. So an elitist evolutionary algorithm can only improve on the best fitness of the unmasked version.
4) compute the best fitness after "z" GA iterations.

In order to verify in the most "honest" way the benefit allowed by this algorithm, results need to be compared with a "unmasked" version of the algorithm allowing z+y*s iterations. In the table below, one can read three output of the algorithms for z=1500, y=300 and s=5. First, for comparative purpose, a simple run of the "unmasked" algorithm is done for 3000 GA iteration, then 5 trials are made of the "masked" version for 300 GA iterations, to be concluded by 1500 GA iterations of the re-establish "unmasked" version, with the best mask extracted from the precedent trials. In such a way, the unmasked version and the "emergent masked" version will be fairly compared following a same number of 3000 GA iterations. In the table below the three outputs are quite representative of the many run we did of this same algorithm.

Focusing on the first output (i.e. the first column), the best fitness attained by the unmasked version after 1500 GA iterations was 5. Then 5 trials of 300 GA iterations were run with 5 different masks. The best fitness, 2, was obtained with the fifth mask: "01457". A final release of 1500 GA iterations was performed by re-establishing the complete coding of the states and the rules, which eventually lead to the optimal rule table. Here the right way to observe the CA turned out to be by exploiting the mask "01457", an observable that was really selected by the system itself. Following this last run, a final perfect adder was obtained for the 5 numbers (fitness = 0). In general, this algorithm leads to the best final fitness we could ever have, with an average of 1.5 after the 5 masks testing and the final 500 iterations instead of 6 by simply using the original unmasked coding during 3000 iterations. A considerable improvement results from the use of this emerging observable of the CA. Results shown in the 2nd and the 3rd columns are also interesting since, despite the mediocre results obtained after 500 iterations with mask "01456" in the first case and "02356" in the second, the following "masked" version considerably improves the final fitness (2 and 1). So in all cases, it seems that the use of the masks constrain the GA search space in a very profitable way.

No masking: fitness = 5	No masking: fitness = 7	No masking: fitness = 7
With masking: 01347 fitness = 7	With masking: 01456 fitness = 7	With masking: 12357 fitness = 7
With masking: 01245 fitness = 4	With masking: 23567 fitness = 10	With masking: 02356 fitness = 6
With masking: 24567 fitness = 6	With masking: 12467 fitness = 9	With masking: 12356 fitness = 8
With masking: 01234 fitness = 6	With masking: 12457 fitness = 10	With masking: 01237 fitness = 10
With masking: 01457 fitness = 2	With masking: 12357 fitness = 9	With masking: 01346 fitness = 7
No masking: 01457 fitness = 0	No masking: 01456 fitness = 2	No masking: 02356 fitness = 1

Conclusions and Related Work

It is necessary here to spend some lines on the closely connected and very influential work of Crutchfield and Mitchell [6][7] who also evolve CA both in an engineering and in an epistemological perspective. Regarding the epistemological impact of their work, a new way of observing the CA in terms of "regular domains", "particles", and "particle interactions" is proposed. To some extent, this new observable can be said to emerge since it provides the system with a better understanding of how the CA performs. Now this is somewhat different from our proposal where this new macro way to observe the CA behaviour is not aiming at a better comprehension of how it performs but straightforwardly at a better performance. Roughly said, the adoption of this new sophisticated and high-level semantics based on "particles and particle interactions" allow to understand why the CA

can classify or synchronize but does not make it synchronize or classify better (indeed the coding and general framework they used was enough to find the optimal CA), whereas the "emergent abstract observable" discussed in this paper aims at improving (here by simply accelerating the search) how the CA can achieve addition. Both works illustrate the necessary condition for a phenomenon to "intrinsic emerge" since, in both cases, additional functionally is provided to the system. Simply, the nature of this functionality turns out to be different, oriented towards a better understanding on the one hand and towards a better performance on the other hand. In contrast with more dynamical phenomena like the game of life "glider", this very small increment in abstraction provided by superposing the mask on the CA is a very elementary structural form of emergence, but what is more relevant here is the way it autonomously appears rather than its definitive nature. Besides, but much more work should be done, by observing the CA dynamics when adding the two initial numbers, it clearly appears that some kind of particle is crossing the space to encounter others of the same kind, and that the adoption of this "mechanical" high level semantics could, here also, clarify the rules of addition evolved by the GA. It would have been even more convincing if, instead of the mask, it is these new reading of CA in terms of particles and particle encounters that would have been autonomously selected to accelerate the discovery of the optimal adder.

Adopting a more engineering perspective, this manner of reducing and constraining the search space by adopting a more coarse grained observation of the system to optimize could be effective for a lot of combinatorial and real optimization applications. Take for instance the Traveling Salesman Problem, a way of grouping the cities in some kind of region and first optimize the path among the regions, then keep the best "regionalization" and, from this new solution, try to optimize the real one, is definitely something to try when the number of cities prohibit any acceptable solution by just staying at the lowest level. Also, this technique is similar to some form of automatic feature selection which is a rather common attitude when having to optimize real functions with many dimensions. An obvious next thing to try will be to put the mask under genetic control to let a population of masks to evolve as part of the whole GA's evolutionary process.

Finally, the idea that adopting some "distorted" way of observing the reality can make the observer more adapted in his environment than by trying too match all details will not surprise researchers interested in the study of perception both correct and illusory. Living systems calibrate their perceptual apparatus not to perfectly match the external reality but to extract what is needed for a better life. We don't see the world as it is but as it fits. The work presented here resonates with this generally accepted view of perceptive systems: Here, the evolving system does not observe the CA with all possible details but just in a way which allows rushing towards the most adapted one. Whenever a functional perspective is adopted in biological systems, Darwinism is just around the corner so that, like in the practical work presented in this paper, an emergent observable should be selected in a biological system on account of the adaptive value it provides this system with.

References

1. Andre, D., Forrest H. Bennet and J.R. Koza. 1996. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In J.R. Koza, D.E. Goldberg, D.B. Fogel and R.L Riolo (eds) – Genetic Programming 1996: Proceedings of the first conference, pp. 3-11, MA. The MIT Press.
2. Baas, N.A. 1994. Emergence, hierarchies, and hyperstructures. In C.G. Langton (Ed.) Artificial Life III. Sante Fe Studies in the Sciences of Complexity, Proc. Vol. XVII. (pp. 515-537). CA: Addison-Wesley.
3. Bedeau, M.A. 1997. Weak emergence. In J. Tomberlin (Ed.) Philosophical perspectives: Mind, causation and world, Vol. 11 (pp. 375-399), MA: Blackwell.
4. Bersini, H. and G. Seront (1992): "In search of a good optimization-evolution crossover" - In Parallel Problem Solving from Nature, 2 - Männer and Manderick (Eds.) - pp. 479 - 488.
5. Cariani, P. 1997. Emergence of new signal-primitives in neural networks. Intellectica, 2, pp. 95-143.
6. Crutchfield, J.P. 1994. Is Anything Ever New? Considering Emergence. In Integrative Themes. G. Cowan, D. Pines and D. Melzner (eds.) Santa Fe Institute Studies in the Sciences of Complexity XIX. Addison-Wesley, Reading , MA.
7. Crutchfield, J.P. and M. Mitchell. 1995. The evolution of emergent computation. Proceedings of the National Academy of Science, 23(92):103. van Leeuwen, J. (ed.): Computer Science Today. Recent Trends and Developments. Lecture Notes in Computer Science, Vol. 1000. Springer-Verlag, Berlin Heidelberg New York.
8. Emmeche, C., S. Koppe and F. Stjernfelt. 1997. Explaining emergence: Towards an ontology of levels. Journal for General Philosophy of Science, 28:83-119.
9. Holland, J.H. 1992. Adaptation in Natural and Artificial System. 2nd edition. Cambridge, Mass. The MIT Press.
10. Kubik, A. 2003. Toward a formalization of Emergence. In Artificial Life 9: pp. 41-65. MIT Press.
11. Packard, N.H. 1988. Adaptation toward the edge of chaos. In J.A.S. Kelso, A.J. Mandell, M.F. Shlesinger (eds.) Dynamic Patterns in Complex Systems, pp. 293-301. Singapore World Scientific.
12. Poundstone, W. 1985 The Recursive Universe. Chicago: Contemporary Books.
13. Rasmussen, S., Baas, N.A., Mayer, B., Nilson, M., & Olegen, M.W. 2002. Ansatz for dynamical hierarchies. Artificial Life, 7, pp. 367-374.
14. Ronald, E.A., Sipper, M. & Capcarrère, M.S. 1999. Design, observation, surprise! A test of emergence. Artificial Life, 5, pp. 225-239.
15. Sipper, M. 1998. Computing with cellular automata: Three cases for nonuniformity. Physical Review E, 57(3).
16. Wolfram, S. 1994. Cellular Automata and Complexity. Addison-Wesley, Reading, MA.