On the Emergence of Macro Spatial Structures in Dissipative Cellular Automata, and its Implications for Agent-based Distributed Computing

Franco Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria Università di Modena e Reggio Emilia
Via Allegri 13 – Reggio Emilia– ITALY
franco.zambonelli@unimo.it

Andrea Roli
Dipartimento di Elettronica Informatica e Sistemistica
Università di Bologna
Viale Risorgimento 2 – 40136 Bologna – ITALY
aroli@deis.unibo.it

ABSTRACT

This paper describes the peculiar behavior observed in a class of cellular automata that we have defined as "dissipative", i.e., cellular automata that are "open" and makes it possible for the environment to influence the evolution of the automata. Peculiar in the dynamic evolution of this class of cellular automata is that stable macro-level spatial structures emerge from local interactions among cells, a behavior that does not emerge when the cellular automaton is "closed", i.e., when the state of a cell is not influenced by the external world. On this basis, the paper discusses the relations of the performed experiments with the area of open distributed computing, and in particular of agent-based distributed computing. The basic intuition is that dissipative cellular automata express characteristics that strongly resembles those of wide-area open distributed systems based on autonomous and situated active components – as agents are. Accordingly, similar sorts of macro-level behaviors are likely to emerge and need to be studied, controlled, and possibly fruitfully exploited.

Keywords: Cellular Automata, Self-organizing Systems, Open Agent Systems, Agent-oriented Software Engineering

1 INTRODUCTION

Autonomy and situatedness are intrinsic characteristics of agents that are more and more pervading modern software systems [Jen00, Zam01]. On the one hand, several software systems already include proactive components capable of autonomous behavior, such as mobile and embedded computer-based devices, that can be assimilated – from a software engineering perspective – to autonomous and proactive agents, and that can be modeled as that. On the other hand, several components of these systems are intrinsically situated in an environment, whether a computational one, e.g. a Web site, or a physical one, e.g., a manufacturing system, and their execution is intrinsically associated with local interactions in this environment.

Agent researchers, as well as some researchers in the "mainstream" software engineering community, recognize that both autonomy and situatedness are effective abstractions for the design and development of complex software systems: *(i)* designing applications around autonomous application components, e.g., agents, rather than increasing complexity, can even simplify application design and development over traditional, component-based and object-based, approaches [Par97]; *(ii)* enforcing locality in interactions and explicitly introducing an environmental abstraction naturally matches the characteristics of open multiagent systems, of Web-based systems, and of embedded systems [Zam01]. However, autonomy and situatedness can also become sources of complexity and of engineering problems. In fact, as the experiments focus of this paper shows, in the presence of autonomous and situated software components, the dynamics of the environment can dramatically affect the global behavior of a software system.

In this paper, we present and discuss a set of experiments that we have performed on a new class of cellular automata that we have defined as *Dissipative Cellular Automata* (DCA). DCA differ from "traditional" cellular automata [Wol94] in two characteristics: while "traditional" cellular automata are composed of cells that interact with each other in a synchronous way and that are influenced in their evolution only by the internal state of the automata themselves, dissipative ones are *asynchronous* and *open*. On the one hand, cells update their status independently of each other, in an "autonomous" way. On the other hand, the automata live dipped in an environment that can directly influence the internal behavior of the automata, as in open systems. In other words, DCA can be considered as a minimalist open agent system and, as that, their dynamic behavior is likely to provide useful insight into the behavior of real-world open agent systems and, more generally, of open distributed software systems.

The reported experiments show that DCA exhibit peculiar interesting behavior. In particular, during the evolution of the DCA, and despite the out-of-equilibrium situation induced by the external environment, stable macro-level spatial structures emerge from local interactions among cells, a behavior that does not emerge when the cellular automaton is synchronous and closed (i.e., when the state of a cell is not influenced by the environment). On this basis, the paper argues that similar sort of macro-level behaviors are likely to emerge as soon as multiagent systems (or likes) will start populating the Internet and our physical spaces, both characterized by their own processes and by intrinsic and unpredictable dynamics. Such behaviors are likely to dramatically influence the overall behavior of our networks at a very large scale. This may require new models, methodologies, and tools, explicitly taking into account the environmental dynamics, and exploiting it during software design and development either defensively, to control its effects on the system, or constructively, as an additional design dimension.

This paper is organized as follows. Section 2 sketches the main characteristics of "traditional" cellular automata. Section 3 introduces the class of dissipative cellular automata, presents some macro-level spatial structures emerged from experiments, and attempts at explaining this behavior. Section 4 discusses the relations between dissipative cellular automata and distributed agent-systems, and analyzes the possible impact on the latter of the performed experiments. Section 5 discusses related works. Section 6 concludes and discusses work in progress.

2 CELLULAR AUTOMATA

Generally speaking, Cellular Automata (CA) are regular lattices of cells, each one being a finite-state automaton. At each iteration, cells update their state depending on a (typically simple) state transition function of their state and of the state of neighboring cells. The scientific interest on CA comes primarily from the fact that, despite the simplicity of local rules, they can show complex global behaviors. In fact, the evolution in time of the system exhibits a variety of dynamic patterns related to the state of the cells in the lattice: fixed configurations of cells always in the same state, periodic configurations, complex structures evolving in time. The global behavior of the CA is determined by the local function and the neighborhood structure chosen.

More formally, a CA is statically defined by a quadruple

$$A = (S , d , V , f),$$

where S is the finite set of possible states a cell can assume, d is the dimension of the automaton, V is the neighborhood structure, and f is the *local transition rule*. The automaton structure is a d-dimensional discrete grid $L=\mathrm{Z}^d$, where Z is the set of integers. Each cell is identified with an array of d components $\mathbf{i}=(\mathrm{i}_1,...,\mathrm{i}_d)\in L$ which represent the coordinates of the cell in the grid. It is generally assumed that the grid is infinite, either not limited or closed to a d-dimensional torus. The state of a cell is expressed as a variable x whose domain is defined by S; and the ordered list of cell states defines the CA *global* state X. The neighborhood structure V defines which cells "influence" any cell. V is defined as a function $V:L\rightarrow\wp(L)$ which maps a cell to a set of cells. The neighborhood structure is regular and isotropic, i.e., V has the same definition for every cell. Usually, V is a subset of the group of translations in L. Finally, the *local transition rule* is a function $f:S^V\rightarrow S$ which maps a configuration of states in a neighborhood to a state. The transition rule defines the future state of a cell depending on the state of its neighboring cells (and, possibly, the state of the cell itself). f is typically the same for each cell (uniform CA).

The quadruple A specifies the "static" characteristics of an automaton. However, the complete description of a CA requires the definition of its *dynamics*, i.e., of the dynamics ruling the update of the state of the CA cells. In general, the dynamics of a CA assumes a discrete time: cells update their state in discrete time steps $\mathrm{t}\in\mathrm{N}$ according to the equation

$$x(\mathbf{i};\mathrm{t}+1) = f[x(\mathbf{i};\mathrm{t}), Y(t)],$$

where $Y(t)=\{x(\mathbf{j};\mathrm{t}) \mid \mathbf{j}\in V(\mathbf{i})\}$, i.e., $Y(t)$ is the set of states of neighboring cells of cell **i** at time t.

The usual definition of CA is with *synchronous dynamics*: cells update their state in parallel at each time step. If we assume a finite number of cells n (which is always the case in practice) and we identify cells with n variables x_i, $\mathrm{i}=1,2,\dots,n$, the global state evolves according to the following equation:

$$\mathrm{X}(\mathrm{t}+1) = \mathrm{F}[\mathrm{X}(\mathrm{t})] = \mathrm{F}[\{x_1(\mathrm{t}), x_2(\mathrm{t})\dots, x_n(\mathrm{t})\}] =$$

$$= \{f[x_1(\mathrm{t}),Y_1(\mathrm{t})], f[x_2(\mathrm{t}), Y_2(\mathrm{t})],\dots, f[x_n(\mathrm{t}), Y_n(\mathrm{t})]\} =$$

$$= \{x_1(\mathrm{t}+1), x_2(\mathrm{t}+1)\dots, x_n(\mathrm{t}+1)\}.$$

The whole system evolution is thus described by the evolution of X(t). Since the transition rule is deterministic and both states and cells are finite, the system will eventually reach a stable state, having reached a fixed point attractor, or will periodically pass through a sequences of state, having reached a *cyclic attractor*.

In this paper, we consider 2-dimensional CA (*NxN* square grids with wraparound borders) with two states. These kinds of CA have been deeply studied and have also a biologic interpretation: cells can be interpreted as alive/dead, depending on their state.

As an example, Figure 1 shows an initial random situation in a 2D CA whose cells can be dead (yellow cells) or alive (red cells). Let us consider the Moore neighborhood structure (the neighbors of a cell are the 8 one defining a 3x3 square around the cell itself) and the following transition rule:

f = *{a died cell gets alive iff it has 2 neighbors alive; a living cells lives iff it has 1 or 2 neighbors alive}*

Once the CA starts to evolve from the initial random situation, the states of all cells synchronously change accordingly to the above rule, and after a transient eventually reaches the final cyclic attractor of which one of the composing global states is shown in Figure 2. Of course, assuming other transition rules, the CA can show different behaviors and reach different basins of attraction. We forward the interested reader to [Wol94] for a systematic analysis and classification of synchronous CA.

3 DISSIPATIVE CELLULAR AUTOMATA

In this section, we introduce a new class of CA that we have defined as "dissipative cellular automata", being characterized by asynchrony and openness, and discuss the peculiar behavior that we have observed.

3.1 Asynchronous Dynamics

Section 2 has concentrated on CA with synchronous dynamics, and a large part of literature is dedicated to the analysis of behavior of synchronous CA [Wol94]. However, synchronous dynamics is hardly representative of real-world phenomena, making it not suited for the modeling and the simulation of those phenomena involving a population of interacting elements, for which asynchronous dynamics have to be introduced.

Accordingly to the most accepted terminology, a CA is *asynchronous* if cells can update their state independently from each other, rather than all together in parallel, according to a dynamics that can be either a step-driven or a time-driven.

In *step-driven* dynamics, a kind of global daemon is introduced, whose job is to choose at each time step one (and only one) cell to update. There are several ways to determine the "sequence" for cells update. For instance, cells can be ordered in a fixed or random sequence, or a cell can be randomly selected at each time steps. Thus, step-driven asynchronous dynamics is characterized by a global transition like the following:

$$\mathrm{X}(\mathrm{t}+1) = \mathrm{F}[\mathrm{X}(\mathrm{t})] = \mathrm{F}[\{x_1(\mathrm{t}), x_2(\mathrm{t})\dots, x_n(\mathrm{t})\}] =$$

$$= \{x_1(\mathrm{t}),x_2(\mathrm{t}),\dots f[x_\mathrm{k}(\mathrm{t}), Y_\mathrm{k}(\mathrm{t})],..., x_n(\mathrm{t})\} =$$

$$= \{x_1(\mathrm{t}), x_2(\mathrm{t}),\dots, x_\mathrm{k}(\mathrm{t}+1),... x_n(\mathrm{t})\}$$

where the cell k that is selected for the transition is determined by the specific kind of selection policy.

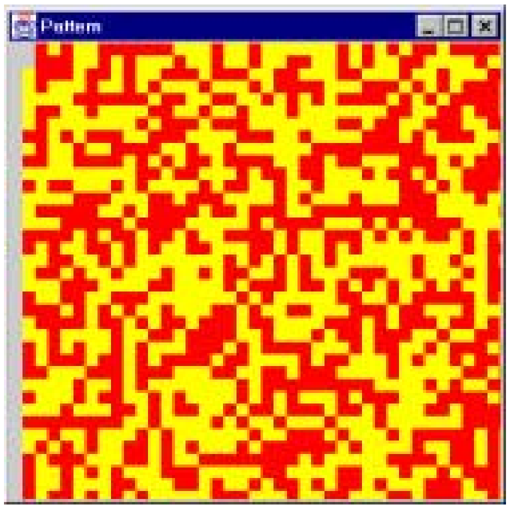


Figure 1: An initial random situation in a 2-D CA

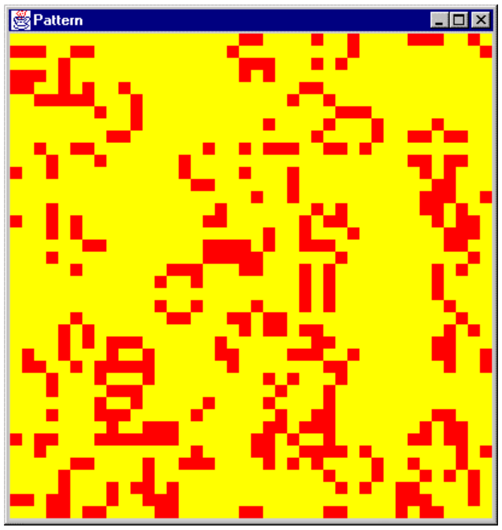


Figure 2: A synchronous CA having reached a global state of a cyclic attractor.

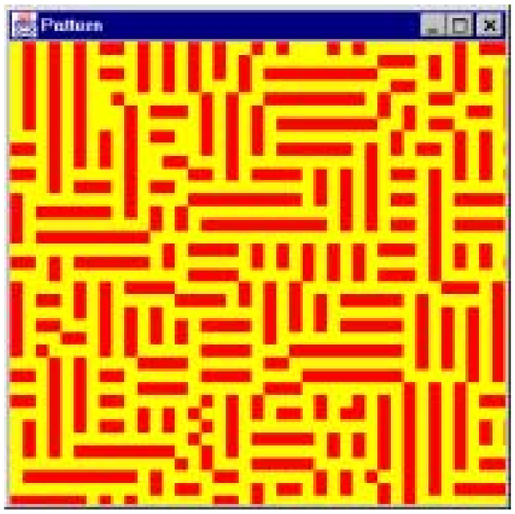


Figure 3: A fixed point reached by an asynchronous CA

In *time-driven* dynamics, each cell is assumed to have an "internal clock" which wakes up the cell and makes it update. This is also the case of more interest to us, in that in open distributed computing agents and processes execute and interact asynchronously accordingly to a local internal clock. Also, time-driven dynamics provides for a more continuous notion of time. The updating signal for a cell can be either deterministic (e.g., every τ time steps) or probabilistic (e.g., the probability that the cell update its state is uniform with a given rate), and the next state of a cell is selected on the basis of the current state of neighboring cells.

In the experiments presented in this paper, CA have an asynchronous time-driven dynamics: at each time, one cell has a uniform probability of rate λ_a to wake up and update its state. The update of a cell has been implemented as atomic and mutually exclusive among neighbors, without preventing non-neighbor cells to update their state concurrently.

In general, it has been observed that the asynchronous CA exhibits behaviors that are very different from the ones of their synchronous counterparts, both in terms of the transient and of the final attractor. Although both the dynamics have the same fixed points [SchR99], i.e., attractors that are fixed points under synchronous dynamics are fixed points also under asynchronous dynamics and vice versa, the basins of attraction can be very different: some of the final attractors reached under asynchronous dynamics are hardly reached under synchronous one.

As an example, Figure 3 shows the fixed point reached by the asynchronous counterpart of the example CA described in Section 2. Under asynchronous regime, this CA usually reaches a fixed-point attractor that its synchronous counterpart has never been observed to be able to reach.

3.2 Openness

CA studied so far are closed systems, as they do not take into account the interaction between the CA and an *environment*. Instead, the new class of CA that we have studied is, in addition to asynchronous, "open", in the sense that the dynamic behavior of the CA can be influenced by the external environment.

From an operative point of view, the openness of the CA implies that some cell can be forced from the external to change its state, independently of the cell having evaluated its state and independently of the transition function (See Figure 4).

By considering a thermodynamic perspective, one can consider this manifestation of the external environment in terms of energy flows: forcing a cell to change its state can be considered as a manifestation of energy flowing into the system and influencing it [NicP89]. This similarity, together with the facts that the activities of the cells are intrinsically asynchronous and that the externally forced changes in the state of cells perturb the CA in an irreversible way**,** made us call this kind of CA as *dissipative cellular automata* (DCA).

From a more formal point of view, a DCA can be considered as:

- $A = (S , d , V , f)$;
- asynchronous step-driven dynamics (with uniform distribution of rate λ_a);
- a perturbation action $\varphi(\alpha,D)$.

where A is the quadruple defining a CA, the dynamics is the one already discussed in Subsection 3.1, and the perturbation action φ is a transition function which acts concurrently with f and can change the state of any of the CA cells to a given state α with some probabilistic distribution D, independently of the current state of the cells and of their neighbors. Specifically, in our experiments with $V=\{0,1\}$, $\alpha=1$ and D is a uniform distribution of rate λ_e.

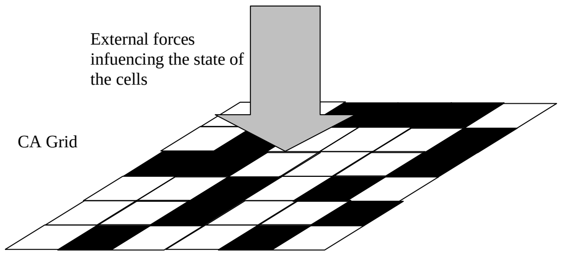


Figure 4: The basic structure of a dissipative cellular automaton

3.3 Emergent Behaviors

The behavior exhibited by DCA is dramatically different from both their synchronous and closed asynchronous counterparts.

In general, when the degree of perturbation (determined by λ_e) is high enough to effectively perturb the internal dynamic of the DCA (determined by the rate of cell updates λ_a) but it is still not prevailing over it so as to make the behavior of the DCA almost random (which happens when λ_e is comparable λ_a), peculiar patterns emerge. The interested reader can refer to the page http://polaris.ing.unimo.it/DCA/ to repeat the experiments on-line.

We have observed that the perturbation on the cells induced by the external – while keeping the system out of equilibrium and making impossible for it to reach any equilibrium situation – makes the DCA develop large scale regular spatial structures. Such structures exhibit long-range correlation between the state of the cells, emerged despite the strictly local and asynchronous transition rules, and breaks the spatial symmetry of the final state. In addition, such structures are stable, despite the continuous perturbing effects of the external environment.

As an example, Figure 5 shows two different patterns emerged from a DCA, both exhibiting stable macro-level spatial structures. For this DCA, the transition rules and the neighborhood structure are the same of the synchronous CA described in Section 2 and f the asynchronous CA described in Subsection 3.1. In both cases, the presence of global-scale patterns – breaking the rotational symmetry of the automata – is evident. By comparing these patterns with the ones observed in the same CA under asynchronous but close dynamics, one can see that openness has provided for making small-scale patterns, emerged from local transition rules, enlarge to the whole CA size. Once this global states has emerged, they are able to re-stabilize autonomously, despite the fact that the perturbing effects tends to modify them.

As another example, Figure 6 shows two typical patterns emerged for a DCA with a neighborhood structure made up of 12 neighbors (the neighbors of a cell are all cells having a maximum distance of 2 from the cell itself) and with the following transition rule:

f = *{a died cell gets alive iff it has 6 neighbors alive; a living cells lives iff it has 3,4,5, or 6 neighbors alive}*

Again it is possible to see large-symmetry breaking patterns emerge, extending to a global scale the local patterns that tends to emerge under asynchronous but closed regime (Figure 7). The patterns are stable despite the continuous perturbing effect of the environment. Moreover, these patterns are stable and robust but they are dynamic. First, the long diagonal stripes in Figure 6 change continuously in their micro-level shape, while maintaining the same global structure. Second, all this stripes translate horizontally at a constant speed in the DCA lattice. This makes the pattern not a fixed point or a simple cyclic attractor, but rather a quasi-periodic one.

3.4 Explaining DCA Dynamics

The phenomenon underlying the behavior of DCA are very similar – if not the same – of the ones determining the emergence of large-scale structures in dissipative systems [NicP89], e.g., in Bénard’s cells. (See Figure 8).

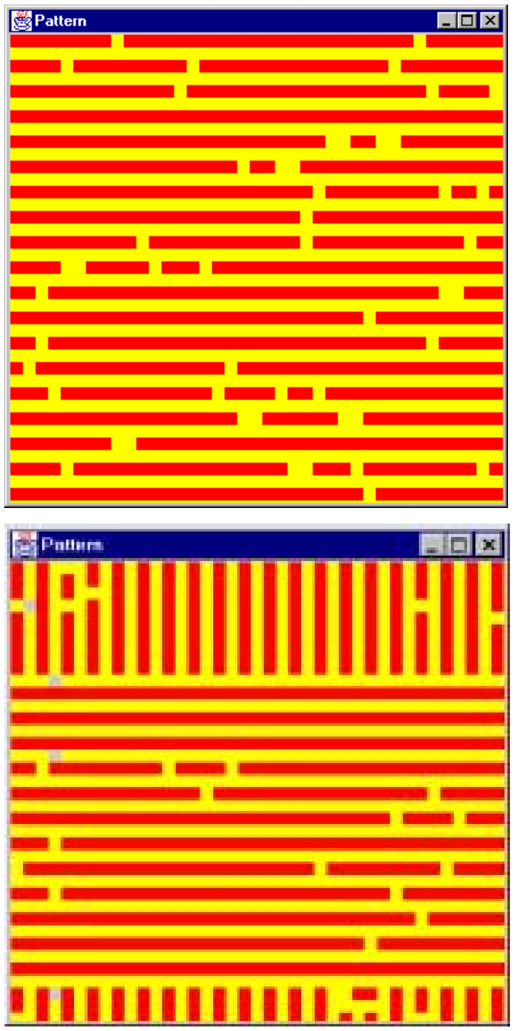
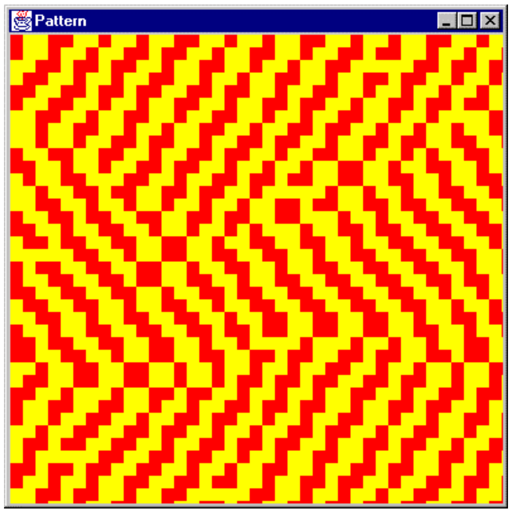


Figure 5: Two Behaviors Evolved in a DCA. Despite the out-of-equilibrium situation forced by the external environment, stable large-scale and symmetry-breaking patterns emerge.



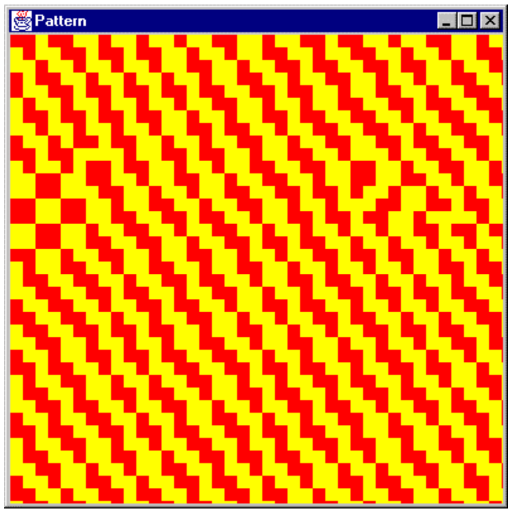


Figure 6: Two different behaviors evolved in a DCA. Large-scale dynamic patterns emerge.

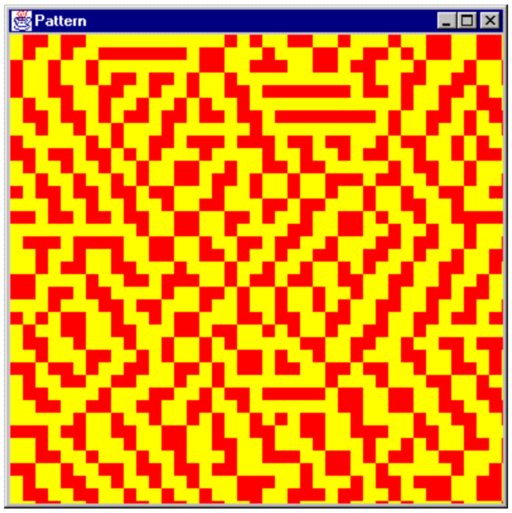


Figure 7: A stabilized situation in an asynchronous closed CA following the same rules of the DCA in Figure .6: no large-scale patterns emerge.

A fluid between two plates is in thermodynamic equilibrium if no thermal energy flows from the external to perturb the equilibrium. In the presence of small differences between the temperatures of the two plates, the thermal energy is still not enough to perturb the fluid, and energy flows between the two plates in the form of thermal diffusion. However, as soon as the temperature gradient reaches a critical point, thermal flow in the fluid starts occurring via convection. This motion does not occur in a disordered way: regular spatial patterns of movement emerge, with wide-range and symmetry breaking correlation among cell movements. This behavior is maintained until the temperature gradient between the two plates become too high, in which case the regular patterns disappear and the fluid motion becomes turbulent.

The behavior of DCA is actually subject to the same phenomenon, where, the temperature gradient between the two plates is substituted by the ratio λ_e/λ_a. When this ratio is 0, the system is in equilibrium, and no perturbation from the external occurs. For very small perturbation, the dynamic behavior of the DCA does not substantially change. As soon as the ratio becomes high enough, the DCA dynamics change and regular spatial patterns appears. For very high ratio, spatial patterns disappear and the DCA dynamics becomes highly disordered. A detailed measurement and a complete quantitative analysis of DCA dynamics are work in progress.

The above similarity suggests that the same causes that determine the behavior of Bénard cells also determine the behavior of DCA.

Without any perturbation, or in the presence of small ones, each autonomous component (a molecule or a DCA cell), acting asynchronously accordingly to strictly local rules, tend to reach a local equilibrium (or a strictly local dynamics), which reflects in a global uniform equilibrium of the whole system.

When the system is kept in a substantial out-of-equilibrium situation, the locally reached equilibrium situations are continuously perturbed, resulting in continuous attempt to locally re-establish equilibrium. This typically ends up with cell groups having found new equilibrium states more robust with regard to the perturbation (or compatible with it). Such stable local patterns start soon dominating and influencing the surrounding, in a sort of enforcing feedback, until a globally coordinated (i.e., with large scale spatial patterns) and stable situation emerges.

When the degree of perturbation is high enough to avoid local stable situations to persist for enough time, they can no longer influence the whole systems, and the situation becomes turbulent.

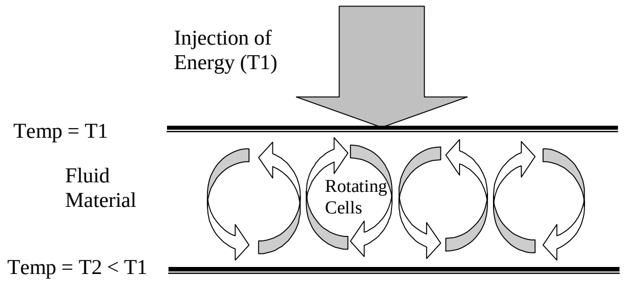


Figure 8: Large-scale patterns in Bénard cells.

4 IMPLICATIONS FOR AGENT-BASED DISTRIBUTED COMPUTING

4.1 Multiagent Systems vs. DCA

There are three characteristics which are typical of distributed multi-agent systems (and that are more and more characterizing all types of software systems) that are reflected in DCA: *autonomy* of agents, *locality* in interactions, *situatedness* in an open and dynamic environment.

Agents are autonomous entities [Jen00], in that their execution is not subject to a global flow of control. Instead, the execution of an agent in a multiagent system may proceed asynchronously, and the agent's state transition occur accordingly to local internal timings. This is actually what happens, because of the adopted time-driven dynamics, in DCA: each state transition in a DCA cell is driven by an internal clock, which is independent from the clock – and the state transitions – of the other DCA cells.

Agents typically execute in the context of a multi-agent organization, and most of the interactions of an agent (causing internal state transitions) occur in the context of that organization [Zam01]. Such abstract concept of locality often reflects also in an actual – physical – locality. In fact, for the sake of scalability and efficiency, multi-agent organizations typically execute in a spatially bounded distributed domain, and wide-area – inter-organizations – interactions are limited as much as possible, and sometimes enabled by making agents move from site to site [CabLZ01]. In DCA, a cell interacts with (that is, can check the state of) only a limited number of other cells in its neighborhood.

Agents are situated entities that live dipped in an environment, whether a computational one, e.g., a Web site, or a physical one, e.g., a room or a manufacturing unit to be controlled. The agent is typically influenced in its execution (i.e., in its state transitions) by what it senses in the environment. In this sense, agents and multi-agent systems are "open systems": the global evolution of a multi-agent system may be influenced by the environment in which it lives. And, in most of the cases, the environment possesses a dynamics that is not controllable or foreseeable. For instance, the computational resources, the data, the services, as well as the other agents to be found on a given Web site cannot be predicted and they are likely to change in time. Analogously, the temperature and lightening condition in a room that an agent is devoted to control may vary dynamically for a number of reasons that cannot be predicted. This sort of openness is the same that we can find in DCA, where the perturbation of the environment, changing the internal state of a cell, can make us consider the cell as situated in an environment whose characteristics dynamically change in an unpredictable way.

Given the above similarities, and given that the characteristics leading to the observed behaviors in DCA are present in most of today's multi-agent systems (and more generally, in modern distributed software systems), there are very good reason to presume that similar strange behavior will be observed as soon a agents will start populating the Internet and our physical environments.

Of course, we are not the first discussing on the possibility of emergence of complex self-organizing behaviors in multi-agent systems. However, most of the studies (apart from a few exceptions [Par01]) have focused on "closed" agent systems, in which the internal dynamics of the systems totally drive its behavior. Instead, we have shown, via a very simple and "minimal" multi-agent system, as a DCA can be considered, that complex non-local behaviors can emerge due to the influence of the environmental dynamics. The impact of this observation in the modeling, engineering, and maintaining of distributed agent systems may be dramatic, delineating a revolutionary change of paradigm in computer science and software engineering [Khu96].

4.2 Defending from Environmental Dynamics

The reported experiments open up the possibility that a software system immersed in a dynamic environment may exhibit behaviors very different from the ones it was programmed for. Of course, this is not desirable and may cause highly damaging effects [ParBS01]. For instance, in the case of a computational Internet pricing system, the emergence of macro-level spatial patterns may produce great price differences in different sites of the planet. In the case of information retrieval applications, this may cause a large amount of available information to be left out from the search, while making the remaining part over-accessed.

A deep re-thinking of the methodologies currently adopted for software design, development, and maintenance is required to avoid such situations to occur, or at least to be able to predict and control them. We do not have solutions at hand, although we can envision some promising directions.

By now, software systems are designed in a mechanical way, component by component, so as to exhibit a specific, deterministic behavior. Such approach immediately fails when the non-determinism intrinsic in environmental dynamics is introduced, and exception handling can only avoid damages to occur, without making the system work effectively. The next challenge is approaching their design in macro-level terms: one should design a system so as to make it exhibit, under a wide range of environmental conditions, the desired global behavior, disregarding if necessary the full understanding of the behavior of its components, and rather trying to understand the behavior of the system as a whole. Moreover, one should precisely characterize the behavior of a system not only in terms of its functionalities, but also in terms of its global behavior depending on the environmental conditions. Possibly, a software system should be designed so as to be able to re-adapt itself dynamically so as to make its internal dynamics contrast the environmental one.

As a consequence of the above approach, software systems will be no longer tested with the goal of finding errors in them, but they will be rather tested with regard to their capability of behaving as needed as a whole, independently of the exact behavior of its component [Huh01], and under the environmental conditions in which the system is expected to operate when released.

It is also important to note that, in most of the cases, a newly deployed software system will execute in an environment where other systems already executes. Thus, the new software system will impact on the environmental condition of the pre-existing systems and, by executing, on their environmental dynamics. Thus, designing and testing a system will not only be devoted to make a software system useful, but also to guarantee that it will not be dangerous to other systems.

We expect theories and models from complex dynamical systems, from modern thermodynamics, as well as from biology, social science, and organizational science, to play a major role in this revolutionary change of paradigm in software engineering, and to become the *sine-qua-non* cultural background for computer scientist and software designers. This cultural background, however, could also help in finding ways to exploit in a constructive the influence of the environmental dynamics.

4.3 Exploiting Environmental Dynamics

Researchers will soon clarifies the mutual dynamic influences between software systems and their environments, and suitable theories, models, methodologies and tools will be developed and mad available to help software engineering and developers. Then, it is very likely that the environmental dynamics will become a useful additional design dimension, other that an enemy to fight.

As a very trivial application example, directly inspired from the visual appearance of the DCA patterns, one could think at “intelligent paintings”. Paintings can be made up of active, radio-enabled, micro-components, able change their colors according to local transition rules, and making it possible to change the color patterns via simple radio-commands perturbing the transition rules and causing a global change in the pattern of a wall. As another example, the possibility of making global patterns emerge from a system relying on local interactions could be exploited so as to enforce global coordination and synchronization in a wide-area system (whether computational or computer-supported) with very low efforts.

More generally, one could think at exploiting the environmental dynamics to control and influence a multi-agent system from “outside the loop” [Ten00], that is, without intervening on the system itself. In a world of continuous computations, where decentralized software systems are always running and cannot be stopped (this is already the case for Internet services and for embedded sensors) changing, maintaining and updating systems by stopping and re-installing them is not the best solution, and it could not be always feasible. For instance, stopping an Internet marketplace for agent-mediated auctions would require stopping accepting new negotiation, making all in-progress negotiation complete, and then updating and testing the new software. The loss in terms of revenues and image would be tremendous. Instead, given the availability of proper models and tools, one could envision the possibility of influence the system without stopping it, simply forcing specific environmental dynamics changing the global behavior of the system so as to make it exhibit the required behavior.

5 RELATED WORKS

5.1 Cellular Automata

Unidimensional and bidimensional CA have been extensively studied [Wol94]. Nowadays, the research area of CA encompasses theoretical studies, as well as applications in the so called *hard* and *soft* sciences.

The hard CA science includes formal studies on computational properties of CA [Sip96, Sip99, ST99], extensions of the simple CA model [Sip96, LumN94, SchR99] and studies on the behavior of CA as complex dynamical systems [Wol94, Bar97]. However, in most of these studies, CA were considered as synchronous and closed systems, for the sake of achieving determinism and predictability in CA’s behavior. Although some work recognize the peculiar and interesting behavior exhibited by asynchronous model [IngB84, LumN94, SchR99], they still missed in identifying the strong influences that the “openness” of the system and the perturbation of the environment can have on the behavior of the CA and on its dynamics.

The soft CA science is splitted in two main branches: applications [Ban98, BanW01] and simulations [Bar97].

In engineering applications, CA are shown to be useful as alternative computing systems for edge detection in digital images, image compression, random number generation [Ban98]. The so called *cellular programming* approach [Sip96, Sip99], where each cell can have its own local transition function (non-uniform CA) and transition functions are assigned and changed by means of an evolutionary algorithm, promises to have useful applications in image recognition, combinatorial optimization problems and evolvable hardware [Sip99]. It is our hope that investigations on DCA may lead to further useful application of cellular-based approach.

In the area of simulation [Bar97], CA are powerful simulation system for biophysical processes and socio-economical phenomena. However, simulation of biophysical processes with classical CA is typically made by considering, in most of the cases, closed systems. The DCA model we have introduced can put these researches forward by providing a suitable framework for the simulation of open biophysical and social systems, other than of computational systems. For instance, on the side of socio-economical processes, simulations with classical CA are viable only under the assumption of that the actor of the simulation, i.e., the CA cells, have perfect knowledge of their own and of their neighbors’ states, and have full control over their own state. Unfortunately, these hypotheses are rarely fulfilled in real-world society and markets. Our DCA approach can be effectively used to model the influence of a dynamic environment and the presence of noise in the process (i.e., limiting both knowledge and control over local states).

Strictly related to the works on CA are those researches on boolean networks [Kau93] and, more recently, on small-world networks [Wat99]. Form a broad perspective, both boolean and small-world networks can be considered as sorts of non-uniform CA with a topology of interconnection that can be described as an undirected graph, typically not regular. These types of networks have found several interesting applications in modeling biological (e.g., genetic) and social (e.g., acquaintance) networks and their dynamics. Still, as in the case of CA, researches related to the influence of a perturbing environment on network dynamics is missing.

5.2 Multi-agent systems

Despite the fact that an agent, accordingly to all established definitions, is an entity situated in an environment, a few researches explicitly focus on the influence of the environment on the behavior of agents and multiagent systems.

Since the origins of distributed artificial intelligence and of multiagent systems researches, a large amount of studies have shown that system in which autonomous components interact with each other in a network, and change their status accordingly to the outcomes of these interactions, can make peculiar global behaviors – whether useful or damaging – emerge [GasB92, HubH93]. Recent examples of these studies may be found in the area of computational markets [KepHG00, GolKS01] and of computational ecosystems [GusF01, Huh01]. However, as anticipated in Section 4, most of these studies focused on the internal dynamics of the system, without taking into account the perturbation of the environment.

Studies in the area of artificial social laws [MosT95] show that global rules constraining the behavior of all the agents in a group can notably influence the dynamic behavior of the group. Analogously, studies adopting an organizational metaphor for the design of multiagent systems [Jen00, CabLZ01, ZamJW01], shows that the definition of global environmental rules to which all agents must obey is very useful toward the effective control of the global multiagent system behavior. For all the above approaches, the basic intuition is that agents, for the very fact of living in an environment (i.e., a society or an organization) are not fully autonomous but, instead, their actions can be constrained by the environment, the same as the state of the cells in DCA can be changed by the perturbation function. However, the above studies exploit such kind of environmental abstractions constructively during the design process, and assume having full control over the environment behavior. Still, these researches typically misses in identifying that agents may live in dynamic environment, where the rules governing their execution and their interactions can change during the evolution of the multiagent systems and can influence their behavior in unpredictable (or simply uncontrollable way).

The importance of the environmental abstraction and of its dynamic in the global behavior of the system is properly attributed in the study and implementation of ant-based multiagent systems [Par97, BonDT99, Par01]. In these systems, very simple agents can indirectly interact with each other in a local way, by putting synthetic pheromones in the environment and by sensing pheromones concentration in a spatially bounded portion of the environment. The environment, by its side, affects interactions with its own dynamics, causing pheromones evaporation or diffusion. Such very simple models may be characterized by self-organization and emergent phenomena that can be useful to achieve difficult goals (swarm intelligence): finding shortest paths, clustering data, etc. The similarities between ant-based multiagent systems and DCA are strong: they both exploit asynchronous components affected in their execution by the environmental dynamics, and both evolve to low-entropy global states [Par01] where long-range correlations are established, generally by means of positive feedback. However, till now, ant-based systems researches have focused on the possibility of "designing" the environment and its dynamics to constructively exploit it, and few researchers focused on the perturbing effects that uncontrollable environmental dynamic can have on the global behavior of a system [ParBS01].

6 CONCLUSIONS AND FUTURE WORKS

This paper has reported the outcomes of a set of experiments performed on a new class of cellular automata, DCA, which are open to the environment and can be perturbed by its dynamics. This experiments have shown that the perturbation makes large-scale symmetry-breaking spatial structures, not observed under closed regime, emerge. Starting from that, the paper has discussed the strong relations between DCA and distributed agent-systems. In particular, the paper has argued that, since distributed agent systems exhibits all of the characteristics of DCA, and in particular openness to the environment, similar sort of spatial structures are likely to make their appearance as soon as agents will start populating the Internet, and are likely to dramatically influence the overall behavior of the network. This requires models, methodologies, and tools, explicitly taking into account the environment and exploiting the environmental dynamics either constructively, as an additional design dimension, or defensively, to prevent and or control the behavior of the system.

The experiments reported in this paper are indeed preliminary, and further work is in progress:

- we are trying to better formalize the concepts of "openness" and of "perturbation", and to better characterize and measure both the degree of perturbation and the degree of order (possibly in thermodynamic terms) of the emergent patterns;
- we are extending our DCA simulation framework so as to study the behavior of networks structures other than the regular ones of DCA, such as small-world and boolean networks, as well as networks with mobile nodes;
- we intend to perform further experiments to evaluate the behavior of DCA under different perturbation regimes and to evaluate the behavior of more complex DCA, i.e., DCA with large set of states, with more complex transition function.

The main objective driving our current research is to make our experiments more and more approximate the characteristics of agent-based distributed scenarios and, eventually, to end up with a powerful simulation environment.

REFERENCES

[Ban98] S. Bandini, R. Serra, F. Suggi Liverani (Eds.). *Proceedings of the 3rd Conference on Cellular Automata for Research and Iindustry*. Springer, 1998.

[BanW01] S. Bandini, T. Worsch, (Eds.). *Proceedings of the 4th Conference on Cellular Automata for Research and Industry*. Springer, 2001.

[Bar97] Y. Bar-Yam. *Dynamics of Complex systems*. Addison-Wesley, 1997.

[BonDT99]E. Bonabeau, M. Dorigo, G. Theraulaz. *Swarm Intelligence. From Natural to Artificial Systems*. Santa Fe Institute - Studies in the Science of Complexity. Oxford University Press, 1999.

[CabLZ01]G. Cabri, L. Leonardi, F. Zambonelli, "Engineering Mobile Agent Applications via Context-Dependent Coordination", *23rd International Conference on Software Engineering*, Toronto (CA), May 2001.

[GasB92] L. Gasser, J. P. Briot, "Object-based Concurent Programming and Distributed Artificial Intelligence", in *Distributed Artificial Intelligence: Theory and Practice*, Kluwer Academic, pp. 81-107, 1992.

[GolKS01]C. V. Goldman, S. Kraus, O. Shehory, "Equilibria Strategies for Selecting Sellers and Satisfying Buyers", *Proc. of the 5th International Workshop on Cooperative Information Agents*, LNAI, No. 2182, pp. 166-177, Sept. 2001.

[GusF01] R. Gustavsson, M. Fredriksson, "Coordination and Control in Computational Ecosystems: A Vision of the Future, in *Coordination of Internet Agents*, A. Omicini et. al (Eds.), Springer Verlag, pp. 443-469, 2001.

[HubH93]B. A. Hubermann, T. Hogg, "The Emergence of Computational Ecosystems", in *SFI Studies in the Science of Complexity*, Vol. V, Addison-Wesley, 1993.

[Huh01] M. Huhns, "Interaction-Oriented Programming", *1st International Workshop on Agent-Oriented Software*

Engineering, LNCS No. 1957, Jan. 2001.

[IngB84] T. E. Ingerson, R. L. Buvel, “Structure in Asynchronous Cellular Automata”, *Physica D*, 10:59-68, 1984.

[Jen00] N. R. Jennings, "On Agent-Based Software Engineering", *Artificial Intelligence*, 117(2), 2000.

[Kau93] S. A. Kauffman. *The origins of order*. Oxford University Press, New York, 1993.

[KepHG00] J. O. Kephart, J. E. Hanson, A. R. Greewald, “Dynamic Pricing by Software Agents”, *Computer Networks*, 32(6): 731-752, May 2000.

[Kuh96] T. Kuhn, *The Structure of Scientific Revolutions*, University of Chicago Press, 3rd Edition, Nov. 1996.

[LumN94] E. D. Lumer, G. Nicolis, “Synchronous Versus 7Asynchronous Dynamics in Spatially Distributed Systems”, *Physica D*, 71:440-452, 1994.

[MosT95] Y. Moses, M. Tenneholtz, “Artificial Social Systems”, *Computers and Artificial Intelligence*, 14(3):533-562, 1995.

[NicP89] G. Nicolis, I. Prigogine, *Exploring Complexity: an Introduction*, W. H. Freeman (NY), 1989.

[ParB01] V. Parunak, S. Brueckner, "Entropy and Self-Organization in Agent Systems", *5th International Conference on Autonomous Agents*, Montreal (CA), May 2001.

[ParBS01] V. Parunak, S. Bruekner, J. Sauter, "ERIM’s Approach to Fine-Grained Agents", *NASA/JPL Workshop on Radical Agent Concepts*, Greenbelt (MD), Sept. 2001.

[Par97] V. Parunak, “Go to the Ant: Engineering Principles from Natural Agent Systems”, *Annals of Operations Research*, 75:69-101, 1997.

[SchR99] B. Schönfisch, A. De Roos, “Synchronous and Asynchronous Updating in Cellular Automata”, *BioSystems*, 51(3):123-143, 1999.

[Sip96] M. Sipper, “Co-evolving Non-Uniform Cellular Automata to Perform Computations”, *Physica D*, 92:193-208, 1996.

[Sip99] M. Sipper. “The Emergence of Cellular Computing”. *IEEE Computer*, 37(7):18-26, July 1999.

[SipT99] M. Sipper, M. Tomassini, “Computation in Artificially Evolved, Non-Uniform Cellular Automata”, *Theoretical Computer Science*, 217(1):81-98, March 1999.

[Ten00] D. Tennenhouse, "Proactive Computing", *Communications of the ACM*, May 2000.

[Weg97] P. Wegner. "Why Interaction is More Powerful than Algorithms", *Communications of the ACM*, 1997.

[Whi97] J. White, “Mobile Agents”, in *Software Agents*, AAAI Press, Menlo Park (CA), pp. 437-472, 1997.

[Wat99] D. Watts, *Small-Worlds*, Princeton University Press, 1999.

[Wol94] S. Wolfram. *Cellular Automata and Complexity*. Addison-Wesley, 1994.

[ZamJW01] F. Zambonelli, N. R. Jennings, M. J. Wooldridge, “Organizational Abstractions for the Analysis and Design of Multi-agent Systems, *1st International Workshop on Agent-Oriented Software Engineering*, LNCS No. 1957, Jan. 2001.

[Zam01] F. Zambonelli, "From Design to Intention: Signs of a Revolution", 2nd Italian Workshop on Objects and Agents, Modena (I), Sept. 2001.