ABOUT NATURE OF EMERGENT BEHAVIOR IN MICRO-SYSTEMS

Sergey Kornienko, Olga Kornienko, Paul Levi
University of Stuttgart
Universitätsstr. 38, D-70569 Stuttgart, Germany

Keywords: Emergent behavior, Micro-robotic systems, Robotic swarm intelligence.

Abstract: Micro-robotic systems have very limited computational and communicating resources on board. However they have a broad spectrum of tasks to be solved. One of approaches to solve these tasks by such capability-restricted systems consists in utilizing emergent properties of many interacting robots. In the presented work we consider the questions what is the emergent behavior in technical systems and how this artificial emergence differs from the natural analogue ? These points are discussed on examples of spatial and functional emergence in a group of "micro-agents".

1 INTRODUCTION

Micro-robotic systems represent a new trend not only in robotics, but also in distributed artificial intelligence. The extreme miniaturization of these systems creates new challenges for a robot's hardware. However the typically "software" concepts of controlling, perceptions and planning get also completely different forms. Because of very small size, these robots do not have sometimes even a CPU-based microcontroller and possess very limited communication bandwidth and range. Despite these limited abilities, a group of robots has to solve a broad spectrum of tasks, as e.g. cleaning, micro-assembling, transportation of micro-objects, collective perception and so on.

A way to achieve the desired collective behavior in a group of micro-robots consists of creating specific swarm-like-behavior, known from the insect world. As shown by natural examples, this emergent kind of behavior is very efficient, flexible and is closely related with collective (or swarm-) intelligence (Camazine et al., 2003). Moreover it does not require complex control systems and allows a large number of independent units to accomplish collectively the common goal. The swarm behavior is widely encountered in natural collective systems, however what is a swarm-like-behavior in robotic systems ? More generally, what is an emergent behavior in technical systems ? Are there advantages of creating this behavior over traditional programming ? Although there is a long discussion in the vast literature on the object (see e.g. (Mataric, 1992)), many points still remain open.

Emergent behavior can not be programmed directly. It is created by specific interactions among subsystems. These, in turn, are determined by local rules, governing behavior of each subsystem. For "insect-standard" problems, like route optimization or collective defence, we can find and adopt the rules from the insect-world (Bonabeau et al., 1999). But for technical activities, like assembling of micro-constructions, we have to derive artificial rules, leading to *desired* emergent behavior. And so we have the following paradox: emergent behavior arises without being programmed, but we are still going to programm it by specific rules. What it does mean "creation of desired emergent behavior" ?

In this paper we consider a problem of emergent behavior in micro-systems within the framework of I-SWARM project (I-Swarm, 2007), paying the main attention to irregularity of technically useful emergent behavior. This irregularity is the distinctive feature of technical systems. We illustrate the emergence with examples such as the formation of spatial groups and the assembly of micro-objects.

2 EMERGENCE IN TECHNICAL SYSTEM

Phenomenon of technical emergence. *"Emergence is a process by which new structures and functions come into being"* (Cariani, 1997). There are several kinds of emergence, e.g. "combinatorial", "creative",

Kornienko S. and Kornienko O. (2004).
ABOUT NATURE OF EMERGENT BEHAVIOR IN MICRO-SYSTEMS.
In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, pages 33-40
DOI: 10.5220/0001142200330040

”thermodynamic” and so on. Generally we refer the emergent property of system to arising of something, being not explicitly programmed. We assume that this arising ”new” possesses some useful properties, otherwise the phenomenon of emergence would not attract so much attention to itself.

Almost all examples of emergence originate from natural systems. Swarm behavior in flocks, insect colonies and shoals is fascinating. However how much this behavior can be applied to precise technological systems of robots, software and mobile agents ? Natural and technical systems are quite different and the questions is whether the swarm behavior in these systems is also different ?

We consider this problem on the maximal simplified example of an assembling of micro-objects. Let we have two different kinds of agents with different abilities and two kinds of objects with different geometry (see fig. 1). The common task, to assemble them into a construction, can be solved only by a cooperation between agents. Usually such a cooperation can be achieved by corresponding scheduling of agents activities. Since we do not preprogram this scheduling, an appearance of cooperation can be thought as an emergent property of this system.

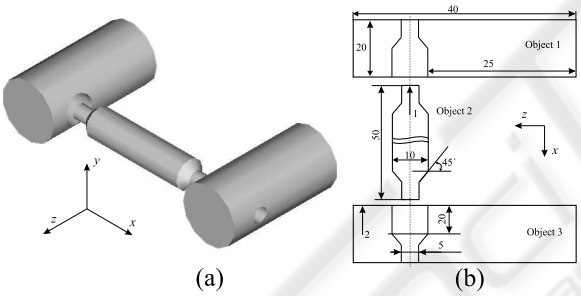

Figure 1: The workpieces to be assembled; **(a)** 3D Representation; **(b)** The x-z section of objects

An assembling of the workpiece should be performed in some specified order, otherwise we do not obtain the desired detail. *Independently of the assembling method, classical or swarm-like, the assembling order should be preserved.* The assembling plan is represented as the Petri Net, shown in fig. 2. The plan consists of 7 steps, shown as the phases p_1-p_7 with the corresponding positions and rotation angles. The order of assembling is following: the phases p_1, p_3 and p_6 can be started in parallel. However, other phases have to be proceeded sequentially. The phase p_7 can be started only if p_5, p_6 are finished. For the phase p_5 we have two cases. The objects Ob^1 and Ob^2 can be assembled if either the object Ob^1 or the object Ob^2 are placed in the required position, $t_5 = \{p_4, p_2\}$ for the first case, $t_5 = \{p_4\}$ for the second one. The restrictions on the order of operations are the global restrictions C_g.

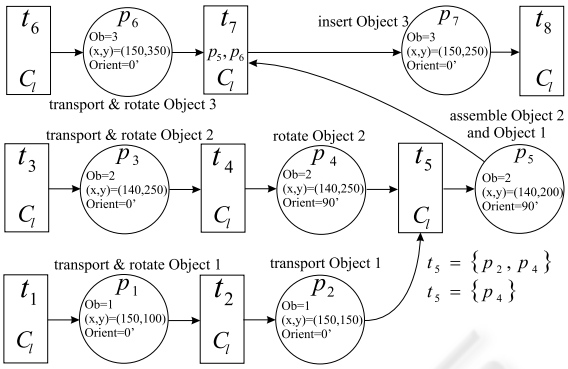

Figure 2: The assembling plan. P_i are phases, where t_i are transitions with the shown conditions (e.g., conditions for t_7 are the satisfaction of local constraints C_l from fig. 3, and the finished phases p_5, p_6)

There are two kinds of agents. The first one Ag^1 can rotate an object, where as the second one Ag^2 can transport an object. Both agents have a ”position” (x, y) in the agent's local coordinate system and have simple 8-directional movement system. Objects also have ”position” (x, y), ”rotation angle” α and ”geometry” (h, l). Each agent observes neighbors in some radius R_{vis}. It can also measure a distance to target and a rotational angle of target (closely to object). In order to simplify the problem, we do not consider collisions between agents and an agent takes an object by placing itself in the geometric origin of an object (x_0, y_0). Each agent reads from the plan only relative distances between objects (position of assembling place is marked by a mark). If an agent starts some activity with an object, it marks this object by putting a number of current phase on the mark (e.g. in the electromagnetic way).

An agent can start transportation or rotation only if its position coincides with the position of an object. Moreover, before starting an activity, an agent has to be sure that the object is not currently processing by other agent or the current activity is not already done by other agent (these problems can be solved by marking). We denote these restrictions as the local restrictions C_l. Activity of each agent can also be represented in the form of Petri Net (see fig. 3). Agents start from random initial positions. Objects are also initially placed in random positions with random angles, but without intersections between objects.

Emergence of cooperation. In simulation, each agent looks for objects in own neighborhood. For the found object, the agent reads the mark and calls the required activities from the locally stored plan. If the local and global restrictions are satisfied, the agent executes the required activities. The local rules of an agent have the following form:

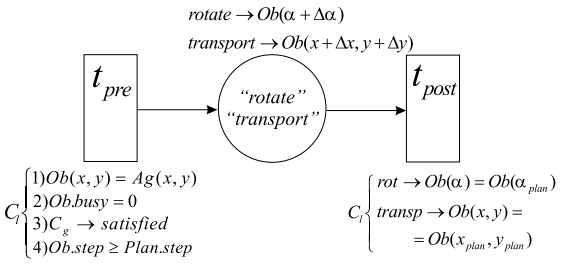

Figure 3: Activity "transport" and "rotate" of agent with local constraints C_l. Activity "move" is called automatically if a position of agent do not coincide with a position of target

```
Ob=look for (visible objects);
read mark (Ob);
if (constraints(Ob)) do (Activity);
```

As already mentioned, agents can start assembling from different initial phases of the plan. In fig. 4 we compare the possible initial phases with the average length (time steps) of assembling. Two gener-

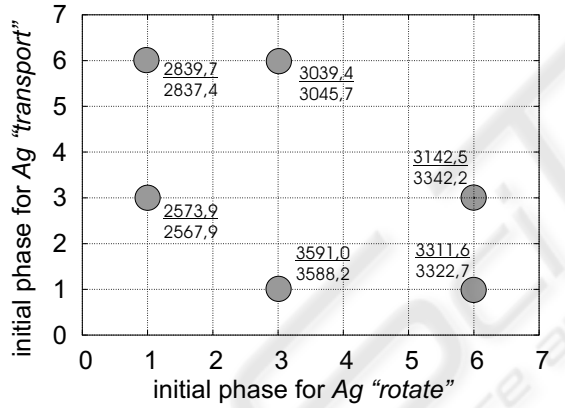

Figure 4: Phases diagram of the Petri-network shown in fig. 2. In the fraction near each initial point the numerator shows an average length of an assembling with $t_5 = \{p_4\}$, the denominator shows an average length of an assembling with $t_5 = \{p_4, p_2\}$. Agents start from random initial conditions,100×100 square,$R_{vis} = 400$, shown is the average result of 10000 simulation's cycles

ated agent-agent cooperation's patterns are shown in figs. 5, 6. We see, the initial order of phases causes completely different cooperation between agents. Therefore we can choose more short assembling by putting additional rules as e.g.

```
at choice ->
   choice phase with smaller number;
```

On this example we can discuss the question why do we need the emergence in technical systems ? *The first advantage of emergent behavior is a simplicity*

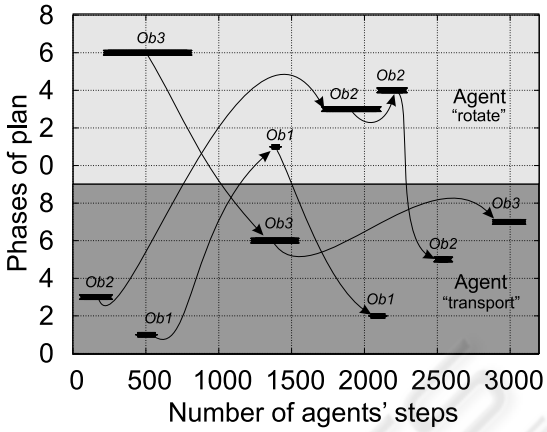

Figure 5: Example of emergent "agent-agent" cooperation, generated by the local rules. Initial phases are $(Ag_1)_{init} = p_6$ and $(Ag_2)_{init} = p_3$

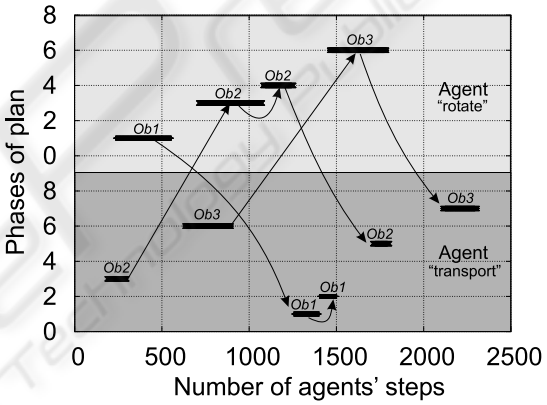

Figure 6: Example of emergent "agent-agent" cooperation, generated by the local rules. Initial phases are $(Ag_1)_{init} = p_1$ and $(Ag_2)_{init} = p_3$

of generating local rules. They can be implemented even in a very restricted hardware. Simplicity of local rules represents important issue for micro-systems. *The second advantage consists in flexibility of generated behavior.* Flexibility means that if some elements of agent's system or of a plan itself will be perturbed, agents can absorb this disturbance and are still able to accomplish their common business. We do not need to reprogram the system every time. To demonstrate it, we perturb positions and rotation angle of all objects. In this way we simulate micro-vibration of mechanical origin. Comparison between unperturbed and perturbed assembling is shown in fig. 7. As seen from this figure, agents can still finish assembling even at very strong positional noise. However they are sensitive to positional-angle noise.

We see, that the cooperation, even in this simple example, emerges without being *preprogrammed.*

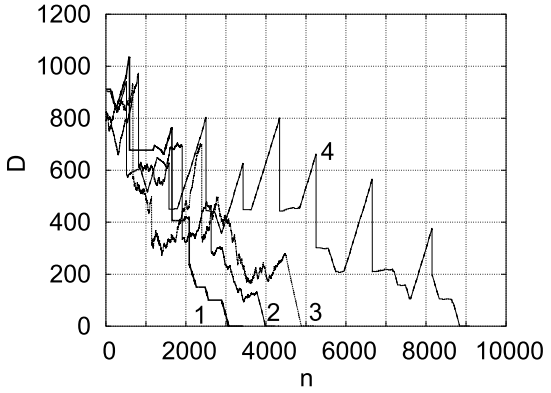

Figure 7: Perturbation of micro-assembling by "micro-vibration", D - difference between the plan and real assembling's state ($\sum$ of all positions and angles), n- number of steps; 1- unperturbed assembling, 2- with perturbed positions of objects (± 1 per simulation's tact); 3- with perturbed positions of objects (± 2 per simulation's tact); 4- perturbed positions and rotation angles of objects (± 1 per 50 simulation's tacts)

Emergence arises because of *interactions* between agents. These interactions, in turn, are determined by *local rules* that govern behavior of each agent. However this kind of emergence differs from natural emergence observed e.g. in biological systems.

Firstly, in technical systems we needed more or less specific spatial or functional emergent behavior. "Specific" means that the behavior should meet predefined requirements. We denote these requirements as "irregularities", because they introduce into normal ("regular") behavioral course irregular components. In the assembling example, they are local and global constraints. We could denote this emergence as constraint or irregular emergence.

Special kind of irregularities are parameters. In the assembling, each operation is *parameterized* by data from the plan. Without knowing these parameters, agent cannot accomplish assembling. There is still open discursion about how many parameters have to be involved in the emergent behavior ? Should the agents e.g. know own neighbors ? Where is a limit of parametrization, after that the behavior is less emergent and more predetermined ? One possible way to answer these questions is to define a compromise between "useful" and "useless" emergent behavior. We can "sacrifice" a "useless" part of collective system, so that to minimize a parametrization of a "useful" part. Generally, parametrization still remains a research point.

Secondly, as observed from the simple assembling example, the emergent behavior can be of different efficiency. In contrast to natural systems, which have to be foremost *reliably*, the technical emergence has to be also *optimal*. Thus, we have to derive such local rules that not only generate the desired emergence, but also optimize it.

3 CONSTRUCTION AND GENERATION OF EMERGENT PATTERNS

Speaking about emergent behavior, we speak primarily about emergent behavioral patterns. This issue, in turn, consists of two following points: construction of desired patterns and generation of desired patterns.

Construction of emergent patterns. This point concerns regularity and irregularity of patterns to be emerged. We can intuitively assume that regular patterns are more "ordered", than irregular patterns. The "order"-degree of a pattern can be associated with the number of rules, required to construct it. Thus, if a pattern is regular we need less rules to construct it, than in case of an irregular pattern. The number of required rules we can estimate by calculating Kolmogorov complexity of corresponding symbolic sequences. To exemplify this idea, let us consider two simple patterns, shown in fig. 8.

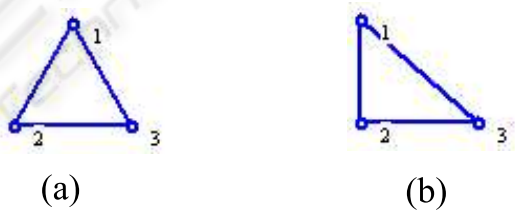

Figure 8: Examples of regular (a) and irregular (b) spatial patterns

We write a pair of distance D (between neighbors j and i) and neighbors i, j as $[D, (j, i)]$. For the shape in fig. 8(a), (b) we have correspondingly the sequences $S_1 = [D, (1, 2)], [D, (2, 3)], [D, (1, 3)], ...$ and $S_2 = [D_1, (1, 2)], [D_1, (2, 3)], [D_2, (1, 3)], ...$ Now we find the rules that can generate S_1 and S_2. For that we use the well-known LZ77 approach (Ziv and Lempel, 1977). The schematic output of LZ77 algorithm in the form $(O, L)C$ (O-offset from current position, L-length, C-chairs) is shown in fig. 9.

We see, that irregularity of the pattern in fig. 9(b) occurs in two ways: **appearance of additional rules** (1st and 2nd cycles) and **parametrization of these rules**. As known, the regular behavior (and in this way local rules) can be derived by optimizing some quantities (e.g. energy consumed by a system) or by some simple principles. Can the local rules in fig. 9 be

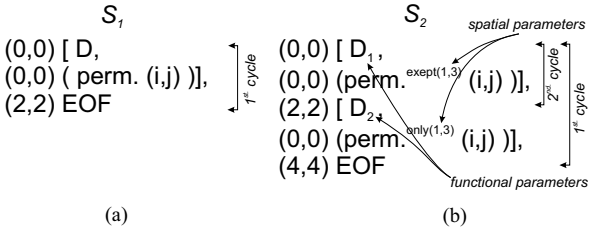

Figure 9: The schematic output of LZ77 algorithm, applied to the sequences S_1 (a) and S_2 (b)

obtained in this way ? We can write such a principle without difficulties for the first sequence S_1:

```
keep up equal distance to all agents;
```

As a result we get an equilateral triangle. If we have to obtain a specific equilateral triangle we have to specify the desired distance. However how to obtain the rules for S_2 in fig. 9(b), especially their parametrization ? In fig. 10 we show more complex irregular pattern. Obviously, this pattern requires essentially more

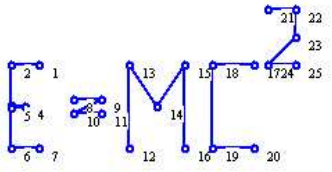

Figure 10: Example of complex irregular spatial pattern

rules and parameters. Could it be generated by some compact evolutional process ? We think it is possible, but the Kolmogorov complexity of generating grammar is much higher, than the irregular generated pattern itself. For the pattern in fig. 10 we know only one evolutional process that can generate it, namely, evolution of human civilization ! Therefore for generating irregular patterns, the irregularities (primarily parametrization) have to be explicitly introduced into rules. Unfortunately, the most of technically useful behavioral patterns are irregular.

The question is whether we can generalize this conclusion for other kinds of behavioral patterns (e.g. functional patterns, like the assembling's plan, shown in fig. 2) ? Here we refer to the genetic programming, namely to the evolving of computer program capable of emergent collective behavior, discussed in the Koza's work (Koza, 1992, p. 340). He considers a group of independent agents with one common goal to consolidate widely dispersed pellets of food into one pile. Agents have behavioral and transportational rules, but initially there is no composition of these rules that allows accomplishing the common task. Koza introduces the fitness function "to minimize the sum of distances between food pellets". This fitness is similar to the evolutional rule that generates the sequence S_1. Performing the GP procedure with this fitness, agents collect the food pellets into one pile. This behavior corresponds to the regular behavioral pattern. However if agent have to collect the food in specific way, or the pile should have specific form (e.g. a storehouse) we have to introduce the parameters that will describe irregularities so that to create the desired emergence. These parameters cannot arise evolutionary, they have to be defined in advance. In this way, the conclusion about parametrization can be expanded to other kinds of behavioral patterns.

Generation of desired emergent patterns. *Emergent behavior arises as a result of self-organization* (SO). Therefore generation of desired emergence consists in creating purposeful self-organization. However such a self-organization that takes place in technical systems. Are there differences between the "artificial" and "natural" self-organization ?

Per definition, "*the self-organization is a process by which global external influences stimulate the start of internal for the system mechanisms, which bring forth the origin of specific structures in it*" (Bushev, 1994, p. 24)). Thus, the often used notion of self-organization is given by the emergence of ordered macroscopic structure in absence of central control. But if we apply this notion to technical systems, many systems will be matched up with this definition. For example, consider organization's local network. Data, e-mails and news are accurately delivered from sender to receiver, printers print different documents, servers operate with clients and so on. All elements of this network remains autonomous, there is no central element, finally, at some control parameters, this system demonstrates either ordered or chaotic behavior. *But this behavior is not a result of self-organization, the order in this system arises in preprogrammed way.*

Could we refer the process of assembling, discussed in the previous section, to self-organization phenomena ? Let us compare this system with natural systems. Firstly, this is the open system influenced from environment. Secondly, modifying the control parameters (e.g the visibility radius R_{vis}), a cooperation between agents becomes ordered. There is no central element, that would tell each agent what is to do. The cooperation is not preprogrammed, it arises from interactions between agents. Therefore, from the general viewpoint, the assembling occurs in the self-organizing way. However this group of agents has remarkable differences with natural systems.

- In the most natural self-organizing systems the control parameters are represented by energy (matter) flow, change of control parameter is given by a change of these quantities. This energy (matter) flow determines interactions among elements in these systems. In artificial systems information is disconnected from energy (matter) flow.

- Interactions among elements in natural self-organizing systems are fixed by chemical, physical or other laws. However the interactions as well as structure of artificial systems can be changed.

In natural systems a dependence between interactions, structure, function, information and control parameters is determined by physical laws introduced into a system by energy flow. This dependence determines effects that we denote as natural self-organization. In artificial systems an energy flow does not influence the system so strongly as in natural systems. Therefore in order to create a purposeful artificial self-organization this dependence has to be defined anew.

Now backwards to the question of what is self-organization in artificial systems. As said above, the artificial self-organization has more degrees of freedom than the natural self-organization. These additional degrees are the structure, rules, transfer functions, information processing, number of elements, control parameters, initial conditions and so on. We represent the structure of these systems in fig. 11.

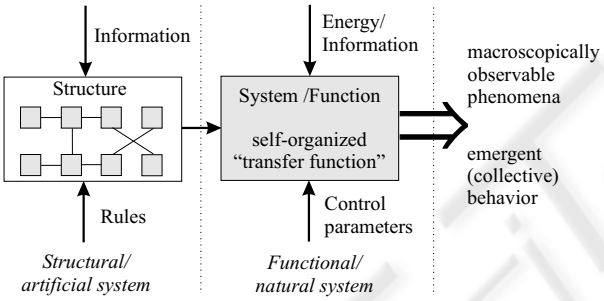

Figure 11: The structure of artificial self-organizing system

We see, that the emergent behavior is macroscopic observable phenomenon, generated by a "transfer function" of a system. Arising of this "transfer function" represents a process of self-organization, which is controlled by control parameters. Changing of control parameters changes "transfer function" and, in turn, macroscopic phenomena. Such a kind of self-organization on the level of functions can be denoted as the functional (or natural) self-organization. *In functional SO interactions among elements are predefined, so that the self-organized "transfer function" is fixed.*

Additional degrees of freedoms in artificial systems appears on the level of structures. Changes of structures modify the "transfer functions", that, in turn, change macroscopic phenomena. The structures consist of basic elements (agents) and interactions among them. Interactions are created by local rules governing basic elements. If there is a mechanism that can systematically change the structure, *the interactions among elements are no longer preprogrammed, they, as well as the self-organized "transfer function", are generated dynamically.* The self-organization created on the level of structures by these "structure-generating-mechanisms" can be denoted as structural self-organization.

Both functional and structural SO phenomena generate emergent behavior. However there is a big difference between them. *The functional SO creates only one emergent behavioral pattern, where as the structural SO generates a cluster of such patterns.*

The functional and structural SO can be created in many different ways. The most often discussed way consists in deriving a set of local rules. There are two strategy to derive them. At the bottom-up strategy, the local rules are first programmed into each agent. This rule-based programming (Roma et al., 1993), originates from the domain of parallel and distributed computing. Generation of these rules is mostly considered in a context of refining sequential program into concurrent one (Back and Sere, 1991).

The general problem of bottom-up approach is that we cannot say in advance, which emergent behavior will be generated by the chosen rules (by analogy with the well-known "three-body problem" from nonlinear dynamics (Arnold(Ed.), 1988)). As pointed out by other authors (e.g. (Darley, 1994)) "*A true emergent phenomenon is one for which the optimal means of prediction is simulation*". The origin of this problem lies in enormous complexity of nonlinearly interacting system. Since the bottom-up derived rules generate only one emergent behavioral pattern (that not necessarily coincides with the desired one), we refer this strategy to the *functional SO phenomenon*.

Another way to derive the desired behavior consists in the top-down strategy, shown in fig. 12. Using

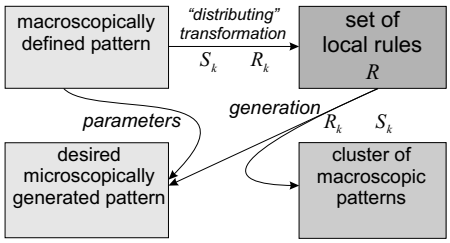

Figure 12: Top-down strategy of derivation of local rules

this strategy, the derivation of local rules starts from definition of a macroscopic pattern Ω. This is a desired collective phenomenon, that the system has to demonstrate. Examples of these patterns are shown in figs. 2, 8, 10. The most of macroscopic patterns can be created without any difficulties. Assume, we have an algorithm, that can decompose an achievement of Ω into n-subtasks Ω_i. We also have a set of agents $\{Ag\}$ with corresponding elementary activities, however so, that they can collectively solve each

of Ω_i. The decomposition algorithm splits up each of Ω_i further, up to elementary agent's activities. Thus, we have $\{\Omega_{i=1...n}^{j=1...m}\}$ sequences of activities, where an agent Ag_k needs m steps to solve Ω_i. Since this algorithm decomposes systematically, we can assume that all agents can solve Ω by executing $\{\Omega_i^j\}$. Remark, that a cooperation between agents arises naturally as the top-down decomposition of common task.

From agent's viewpoint, each agent Ag_k has a sequence of activities $S_k = \{\Omega_1, \Omega_2, ..., \Omega_m\}$. Now, calculating Kolmogorov complexity of sequence S_k (finding the smallest grammar (Charikar et al., 2002)), we can derive local rules R_k that can generate S_k. The set of these rules $\{R\}$ defines a cooperation between agents that allows the agents' group cooperatively to solve the common task Ω. Such a decomposition approach (algorithm of symbolic task decomposition - ASTD) is described in (Kornienko et al., 2004b) and the whole rule-derivation procedure in (Kornienko et al., 2004a).

Remark, that the set of local rules $\{R\}$ generates not only one desired pattern. For example, the assembling rules, shown in sec. 2, can generate an arbitrary assembling process of this type. The specific assembling of the workpiece in fig. 1 arises by parametrization of these rules by data from the plan in fig 2. Therefore we associate the top-down strategy of rule derivation with the structural self-organization. In turn, the structural SO phenomenon can be separated into rule-, parameter- and information-based approaches. Generally, investigation of structural self-organization represent also the point of further investigation.

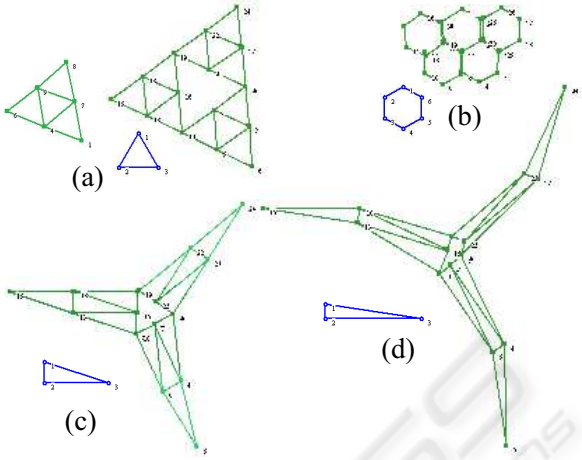

Figure 13: Examples of scaled spatial formations, built dynamically by agents. Small shapes represent basic structural elements of corresponding formations

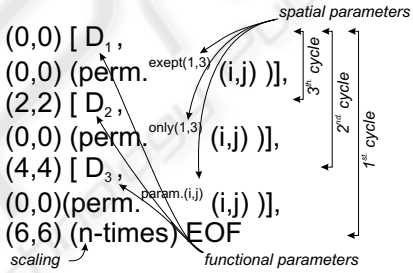

Figure 14: Appearance of rules' hierarchy at the scaling, shown is the schematic output of LZ77 algorithm

4 SCALABILITY OF EMERGENT BEHAVIOR

As known from natural systems, emergent behavior is scalable, the number of participants can be increased and decreased without essential change of behavior's features. However investigating scalability in technical systems we encounter two following issues.

Appearance of rules hierarchy.

If the desired emergent behavior is regular, the scaling does not represent any problems. However if the desired pattern contains irregularities, we need additional rules that describe scalability of irregularities. These irregularities are nonsymmetric form of shapes, specific connections between basic elements (see fig. 13) and so on. As a result, a hierarchy of rules appears (see fig. 14). The more irregularities will be inserted into the scaled pattern, the more hierarchical rules need to be introduced into each agent. There are two main problems of such a rules' hierarchy. The first one consists in a close connection with a size of a group. At really large group there arises a large hierarchy of rules, so that hardware abilities of real micro-robots can be quickly exhausted. The second problem is rules' preconditions. Each behavioral rule has a precondition and a postcondition, as shown in fig. 3. If there arise many hierarchical rules, then there arise also many hierarchical preconditions. At each step, an agent calculates all these preconditions in order to choose the next rule. It consumes time and computational resources. Generally, this problem is also known in other robotic scenarios, e.g. soccer-playing robots in RoboCup. However, for micro-robots, this can have essentially more grievous consequences. Thus irregularities of emergent behavioral patterns represents a serious obstacle, especially in large groups. A strategy to get round this problem consists in finding a compromise between "useful" and "useless" emergency, as mentioned in sec. 2.

Change of collective strategy.

At the scaling, the group can undergo a change of collective strategy. To demonstrate this effect, we put additional "bottom-up" cooperation rule into an as-

sembling rules, discussed in sec. 2:

```
I'm Ag_i;
if (Ag_j=take the same Ob as I){
Ob -> to Ag with smaller Dist. to it;}
```

In fig. 15 we show the comparison between these "bottom-up" and "top-down" rules. For small num-

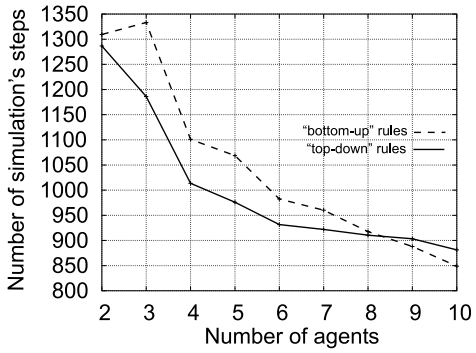

Figure 15: Comparison between the "bottom-up" and "top-down" rules. Agents start from random initial conditions,100×100 square,$R_{vis} = 400$, shown is the average result of 100000 simulation's cycles

ber of agents, the "top-down" rules are more efficient. However, if this number grows, the "top-down" rules becomes less efficient. At some turn-over-threshold, the group changes the collective strategy and the "old" rules can not guarantee any more the achievement of desired emergence. Therefore this effect, and especially a drift of the turn-over-threshold, has to be taken into account at the top-down design of local rules.

5 CONCLUSION

In this paper we have discussed several aspects of desired emergent behavior in technical micro-systems. As shown, technically useful emergence differs from natural emergence in several points, the most important is an appearance of irregularities. The treatment of irregularities concerns coalition formation, constructions of spatial and functional groups, planning and so on. Especially serious problem arises at scaling emergent behavior. Generally, a treatment of irregularities represents a point of further research.

Acknowledgment. The presented work is made in the framework of SFB 467 "Transformable Business Structures for Multiple-Variant Series Production" (supported by the German Research Foundation) as well as EU-Project "Intelligent Small World Autonomous Robots for Micro-manipulation" (I-Swarm).

REFERENCES

Arnold(Ed.), V. (1988). *Dynamical systems III*. Springer Verlag, Berlin, Heidelberg, New York.

Back, R. J. R. and Sere, K. (1991). Stepwise refinement of action systems. *Structured Programming*, 12:17–30.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. Oxford University Press, New York.

Bushev, M. (1994). *Synergetics: chaos, order, self-organization*. World Scientific Publisher.

Camazine, S., Deneubourg, J.-L., Franks, N., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2003). *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ, USA.

Cariani, P. (1997). Emergence of new signal-primitives in neural networks. *Intellectica*, 2:95–143.

Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Rasala, A., Sahai, A., and Shelat, A. (2002). Approximating the smallest grammar: Kolmogorov complexity in natural models. In *Proc. of the 34th ACM symposium on Theory of computing*, pages 792–801. ACM Press.

Darley, V. (1994). Emergent phenomena and complexity. In *Proc. of Alive IV Workshop*, Cambridge, MA.

I-Swarm (2003-2007). *I-Swarm: Intelligent Small World Autonomous Robots for Micro-manipulation*. European Union 6th Framework Programme Project No FP6-2002-IST-1.

Kornienko, S., Kornienko, O., and Levi, P. (2004a). Generation of desired emergent behavior in swarm of microrobots. In *Proc. of the 16th European Conf. on AI (ECAI 2004), Valencia, Spain*.

Kornienko, S., Kornienko, O., and Levi, P. (2004b). Multiagent repairer of damaged process plans in manufacturing environment. In *Proc. of the 8th Conf. on Intelligent Autonomous Systems (IAS-8), Amsterdam, NL*, pages 485–494.

Koza, J. (1992). *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, Massacgusetts, London, England.

Mataric, M. (1992). Designing emergent behaviors: From local interactions to collective intelligence. In J-A. Meyer, H. R. and S. Wilson, e., editors, *Proc. of the 2nd Int. Conference on Simulation of Adaptive Behavior (SAB-92)*, pages 432–441. MIT Press.

Roma, G.-C., Gamble, R. F., and Ball, W. E. (1993). Formal derivation of rule-based programs. *IEEE Trans. Softw. Eng.*, 19(3):277–296.

Ziv, J. and Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343.