Computing and Informatics, Vol. 24, 2005, 617–627

ON COMPUTATIONAL STUDY OF EMBODIMENT: SOME REMARKS AND AN EXAMPLE

Jozef Kelemen

Institute of Computer Science, Silesian University
746 01 Opava, Czech Republic
and Gratex International
821 09 Bratislava, Slovakia
e-mail: kelemen@fpf.slu.cz

Revised manuscript received 12 October 2005

Abstract. Eco-grammar (EG-) systems are proposed as an example of a suitable formal framework for the study of some of the computationally relevant properties of the behavior of collections of embodied agents – called herds in this article – sharing a common environment, and acting in it in simple ways.

Keywords: Computatrion, agent, embodiment, emergence, grammar system

1 INTRODUCTION

Some of the specialists, especially some of those working in the fields of cognitive science, artificial intelligence (AI), and advanced robotics, argue that the source of problems with discovering more adequate and effective ways how to construct (especially how to program) machines in order to provide their continuous functioning in dynamically changing environments consists in the *embodiment* of systems – the phenomenon which remained almost completely ignored in our recent formal computational models. The traditional *mind-body problem* of philosophers and cognitive scientists [7] as well as the actual *software-hardware problem* of computer and robot programmers [1] are from the perspective of embodiment in certain sense identical.

The core of the problem of embodiment consists, according to [14], in the fact that since symbols are abstract entities, computations cannot be performed on them, but have to be mediated through something physical (like organic bodies of living

beings or inorganic bodies of machines) that can be manipulated by some physical operations which correspond systematically to the ones performed during the abstract computational processes over abstract symbols. Moreover, because of the same reasons, the symbols themselves must be represented as physical entities in certain ways. These entities are then manipulated by the above-mentioned physical processes and the results of manipulations are reinterpreted as the results of abstract computations.

The just described *abstract-physical dichotomy* remains unmentioned at all in our theories so far, with all of the consequences of this ignorance; cf. e.g. [15]. But when we concentrate to build embodied systems acting in a physically real, dynamic, usually only hardly predictable environments, we are confronted with the question how the abstract and the physical is interrelated and how this interrelation influences the behavior of our robots, for instance. This is the core of the *problem of embodiment* (at least for the purposes of this paper) – the problem which is highly topical e.g. because of effective construction of different physically embodied autonomous agents. Unfortunately, we have no effective enough tools at hand to study such systems productively, with required theoretical rigor, and from a computationalistic perspective.

In this paper, we will provide a particular example of how at least some of computationally relevant questions concerning embodied agents may be approached from the position of a well-elaborated theoretical (formalized) computational perspective. In particular, we will be interested in situations when several (artificially created) agents are situated and execute tasks in real physical environments. In such a case the agents are faced with objects with real physical properties existing and acting in real time scales. Very hard problems appearing in such situations in the traditional AI research were pointed out first – from very different positions and with very different consequences – by M. Minsky [10] and R. Brooks [1].

Brooks in his concept of the so called novel AI emphasizes the principal role of agents reactivity as a necessary condition of their rationality, while Minsky stresses the principle of decentralization and organization of simplest agents (protospecialists) into more complex ones (into agencies) and presupposes that an agency may play the role of a simple agent in a more complex agency. Both of these positions might be – according our conviction – combined into one approach. The unifying idea behind this approach is based on two basic items:

1. to emphasize the role of as direct as possible interaction of the cognitive systems with their environments at least at the lowest level of sensing and acting, and
2. to exploit the power of organization and of the emergence in lowest levels in order to receive more complex behaviors in highest levels.

The above-mentioned emphases lead us to realize the principal difference between *implementation* of our ideas on how cognitive processes run in natural systems and how they may run in artificial ones, and between *embodiment* of our ideas as artificially constructed agents equipped with sensors (providing signals for them),

with processors (for signal processing and perhaps for computing the decisions), and with actuators (for making changes in their real, dynamic, and noisy environments).

We will focus to the situation, when – for instance because of their embodiment – not all but only some of agents of a multi-agent system are able to act in the shared environment, and in which they are active in a very simple way which requires no addressed interactions, no direct communication, no coordination of actions of several agents, etc. Collections of agents of just sketched type look like herds in the nature. So we will call them *herds* in this article.

The question is whether we are able to construct certain formal theories which might reflect at least some of the properties of the possible behaviors with formal rigor – e.g. the *computational power* – of herds. We will sketch a way how to deal with this sub-problem in the theoretical framework of the so called *eco-grammar* (or *EG* for short) *systems*.

2 THE TRADITIONAL VIEW OF COMPUTATION

As we have mentioned elsewhere [8], according the traditional understanding of computation we can recognize any computing device as an *externally passive entity* whose internal activity is based strictly on activities of a finite number of externally passive components with predefined message passing and on transformation possibilities of this entity.

Thanks to the internal activities of components and their addressed communication the whole computing system transforms the inputs provided to it from certain environment into required outputs. This activity – if it satisfies a dozen of previously well-specified requirements – is interpreted as a computation in the traditional sense developed during the modern history of computing which started in the 30ties of the 20^{th} century with the definition and first studies of (abstract devices equivalent with) the Turing machine.

The Turing machine working in an environment gets its input in advance at the beginning of its work, and outputs the result to the environment at the end of its activity. During the computation, the environment is – from the perspective of the Turing machine – completely passive. Computing and computation are understood as specific processes which reflect the procedural side of function defined in mathematics declaratively as a specific type of relations.

While the function *declares* a specific relation between variables and values in a set theoretic sense (to definition of a function coincides with a defining a suitable subset of the Cartesian product of its domain of variables and domain of its values), the traditional view of a computation (of a function) is *procedural* one: a computation defines a function by means of specifying a step-by-step process of elementary computable steps which transform the given input variable to a corresponding output value (of the corresponding function).

The central problems of (theoretical) computer science originated from the point of view of the just described traditional paradigm of computing are related with the

possibility, description, execution, and the effectiveness of an idealized rule governed by *algorithmic* transformation of input data into the desired outputs.

Inside the just sketched picture of the traditional understanding of any imaginable process which we identify as a computation, in other words the general property of computability – or the (partial) recursiveness (of mathematically defined functions) – is derived from the computing power of the Turing machine. This is the core idea of the so-called *Church-Turing thesis*, which, in a more (but not completely!) precise formulation, states the Turing machine, logics, Church's lambda calculus, algorithmic computing, and the generative capacity of centralized rule-based systems (more precisely the Chomsky-type formal grammars) as equivalent universal machineries for solving computational problems; cf. [18].

3 WHY TO CHANGE THE TRADITION?

In the present time there are strong efforts to prove that the notion of computation might be enlarged beyond the traditional boundaries defined by the Turing machine and the concept of the Turing-computability. In [2] it is proposed to call algorithms and automata that are more powerful than Turing machines as *super-recursive*, and computations that cannot be realized or simulated by Turing machines as *hyper-computations*. In our following consideration on the possible views of computation we will respect this proposal.

Another possibility of viewing systems as computing devices consists in considering a computing device as an *externally active entity* perceiving its dynamic (might be hardly predictable, noisy, or completely unpredictable) outer environment, and acting in it continuously according the perceived stimuli and the own inner rules governing the behavior of the system in order to complete given tasks.

This is the core idea of the third period of the history of modern computing when the individual behaviors of more or less freely cooperating and communicating interacting processors result in a behavior interpretable as a solution of a given problem. The interactivity, as stated in [3] in connection with the analysis of the computational power of the Turing machine *coupled* with its environment, or with the same device appearing as the *interactive* Turing machine in [17] leads to the hyper-computational power of the interacting in the Turing sense computationally universal devices.

The activity of the above mentioned type of systems is based on their own coupling of sensed data with appropriate acts performed in their environment, or on the activities of individually autonomous components forming these systems, and communicating (directly or indirectly) with other components forming them. Systems of this type are usually called *agents*, and the structures formed by these agents are called *multi-agent systems*. In [8] we called the emerging new paradigm of considering collections of such kind of autonomous "open" systems as computing devices instead of the isolated ones as the *agent paradigm* of computing.

Interactions of agents with other agents and with their (dynamically changing, unpredictable, noisy, etc.) external environment during their activities in it are a real promise how to enlarge computational power of systems; cf. e.g. [17]. In general, interactions inside a multi-agent system involve the external word and the activities of individual agents into the behavior (interpreted as a computation) of the whole system *during* the computation (rather than *before* and *after*, as it is in the case of the traditional algorithms) which may lead to the computations that cannot be carried out by a Turing machine as stated in [6].

So, agents and multi-agent systems might be considered as very powerful computational devices and may contribute with many innovative concepts to our traditional picture of the (theoretical) computer science and engineering.

An important dimension of the agent paradigm consists in considering agents not only as products of the development of computer programming techniques and as innovative tools for computer use, but also as products of development of electrical, mechanical, and computer engineering, as electro-mechanical (usually computer guided) devices for automation of different physical processes – as real autonomous machines which do physical (mechanical) work. From such a point of view, as we have mentioned already, there exists an important difference between real computers and the abstract Turing machine.

For instance, in [15] it is stated, that computers, as built and used, are the result of a convergence of the development of machine- and electrical engineering, and of the progress in understanding computations as processes of performing actions on symbols as the Turing machine do that.

Expressing Sloman's observation in our terminology, real computers as well as real agents – (artificially) intelligent systems, especially the cognitive robots – are entities which cannot be divided into their hardware and software parts without missing something fundamental (might be something which emerges) from the functioning of their parts. According to [15], this difference makes computers useful, but Turing machines irrelevant for AI research, for instance.

These two dimensions of agents – interaction with dynamically changing environment and embodiment – converge into a new understanding of machines as embodied, autonomously sensing, acting and deliberating agents – into the form of *robots*.

The above-mentioned difference, the properties such as the autonomy and continuity of machines behavior, the relevance of embodiment, and other physical constrains and limitations (especially the problematic concept of infinity with respect to their behavior), the importance of communication between individually independent, autonomous computational units in order to achieve common goals (intentionally or as an emergent effect of their co-existence in a shared environment), etc. seems to be crucial for embodied systems like robots [11].

Many computational processes in robots processors run continuously and autonomously in different types of environments. Good examples are computing processes running in autonomous mobile robots. When – for instance – a collision avoidance module is programmed, its role is to process the input sensor data conti-

nuously during the robot mission into the data manipulating with robots actuators in order to avoid obstacles in robots environments.

Of course, all the programs of a robot may be decomposed into the set of interrelated programs of traditional type. However, this type of reduction does not contribute to the solution of the problem of collision avoidance at all! Instead of particular programs considered as translation of mathematical functions into some more procedural languages we must think in terms of autonomy and continuity of functioning of systems modules based on their ability to sense the environment and act in it, and on their massive interactions.

In order to apply this new experience in modeling complicated systems (e.g. in economics, sociology, biology, robotics, etc.), the following methodological experience seems to be important: Instead of the necessity to aggregate specific particular data on individual objects as the basis for (mathematical) modeling, the agent paradigm provides a tool for model each individual behavior separately and then study the emerging behavior of the society of these individuals. This methodology is present in many present day experiments with biological, ecological, economic, electro-mechanical (robotic) or conceptual societies of agents.

4 THE COMPUTATIONAL POWER OF HERDS

In the following, the interactions of agents of systems will be in the center of our attention. We will sketch the influence of collections of individually autonomous agents with traditional computing power to the computing power of the whole system set up from these agents, considering the activity of the whole system as a *computation.*

rom computational point of view an appropriate sub-problem of the above-described problem of embodiment consists in rigorous specification of the computational character of results of interaction of the rule governed algorithmic symbol-manipulating processes which run inside the agents which interact with their dynamic environments. Usually we are interested in as precise as possible knowledge of the behavior of an agent or of a multi-agent system in its environment despite of the fact that we have no complete knowledge of the behavior of the environment.

The solution of this problem is twofold: We may study the *possibility* of performing a specified type of behavior thanks to the agents behaviors under the conditions which we put to the behavior of the environment. To solve this type of problem is – in certain extent – the traditional role of theoretical computer science. Another possibility is to concentrate on the *feasibility* (of course, it will be necessary to define rigorously what we mean by feasibility in our considerations). We will deal with the first type of problems using the theoretical framework of the so-called eco-grammar (EG-) systems.

According to [4], an *eco-grammar system* Σ consists, roughly speaking, of

- a finite alphabet V,
- a fixed number (say n) of agents, and evolving according set of rules P_1, P_2, ..., P_n applied in a parallel way as it is usual in L-systems [13], and of

- an environment of the form of a finite string over V (the states of the environment are described by strings of symbols w_E , the initial one by w_0).

The rules of agents depend, in general, on the state (on the just existing form of the string) of the environment. The agents act in commonly shared environment by sets of sequential rewriting rules R_1, R_2, ..., R_n.

The environment itself evolves according a set P_E of rewriting rules applied in parallel as in L systems. The model is schematically depicted in Figure 1.

The evolution rules of the environment are independent on agents' states and of the state of the environment itself. The agents' actions have priority over the evolution rules of the environment. In a given time unit, exactly those symbols of the environment that are not affected by the action of any agent are rewritten.

In the EG-systems we assume the existence of the so called *universal clock* that marks time units, the same for all agents and for the environment, and according to which the evolution of the agents and of the environment is considered.

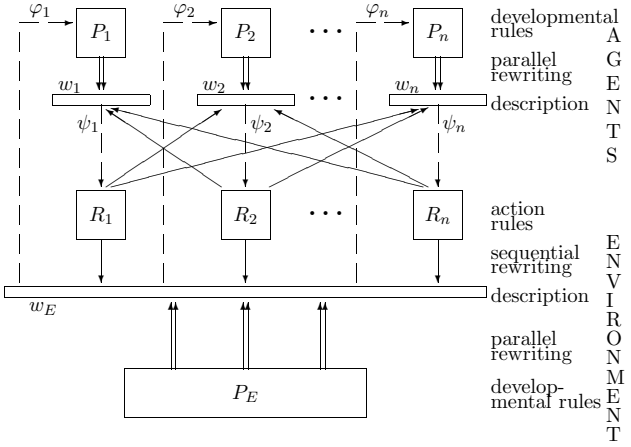

Fig. 1. A schematic view of a traditional EG-system [4]

In [5] a special variant of EG-systems has been proposed in which agents are grouped into subsets of the set of all agents – into the so-called *teams* – with fixed number of members. The idea was to express in the model the embodiment of agents in certain way, especially in this case through limitation of active agents by some space-requirement put on their activities. However, the term *herds* instead of teams seems to be more adequate, because of the lack of direct cooperating and communicating between the agents sharing the common environment and acting in it individually, only.

In [16], where – similarly as in [5] – a variant of EG-systems without internal states of agents is studied, the fixed number of members proposed in [5] is replaced by a dynamically changing number of agents in teams (herds). As the mechanism of reconfiguration, a function, say f, is defined on the set N of integers with values in the

set $\{0, 1, 2, \ldots, n\}$ (where n is the number of agents in the corresponding EG-system) in order to define the number of agents in teams (herds): For the i^{th} step of the work of the given EG-system, the function f relates to a number $f(i) \in \{0, 1, 2, \ldots, n\}$. The subset of the set of all agents of thus EG-system of the cardinality $f(i)$ is then selected for executing the next derivation step of the EG system working with Wätjen-type teams. Wätjen [16] proved, roughly speaking, that there exist EG-systems such that if f is (in the traditional sense) non-recursive function, then the corresponding EG-system generates a non-recursive (in fact a super-recursive) language.

The language is defined in the case of Wätjen's type EG-systems, say Σ, using in each derivation step only agents from the corresponding subset of the cardinality $f(i)$ of the set of agents of Σ, thus:

$$L(\Sigma, f) = \{v : w_0 \Rightarrow^{f(1)} w_1 \Rightarrow^{f(2)} \ldots w_r \Rightarrow^{f(r)} = v, r \in N, w_0 \ldots w_r \in V^*\}.$$

The proof is given in [16] given by contradiction. A recursive language is generated by a special EG-system using arbitrary computable function f. Wätjen uses the EG-system

$$\Sigma = (V, P_E, R_1, R_2, \ldots, R_n, w_E)$$

where

$$V = \{a, b, b_1, b_2, \ldots, b_n\},$$

$$P_E = \{a \to a^2, b \to b^2\} \cup \{b_i \to b^2 | i = 1, 2, \ldots, n\},$$

$$Ri = \{b \to bb_i\}, 1 \leq i \leq n,$$

$$w_0 = a^2b^2n + 3m, m \in N.$$

This EG-system generates the following language:

$$L(\Sigma, f) = \{a^2b^2n + 3m\} \cup$$

$$\cup \bigcup_{k \in N} \left(\bigcup_{\substack{1 \leq i_1 \ldots i_{f(k)} \leq n \\ i_j \neq i_{j'}, j \neq j', 1 \leq j, j' \leq f(k)}} \left\{ a^{2^{k+1}} \right\} perm \left(bb_{i_1}, \ldots, bb_{i_{f(k)}}, \underbrace{b^2, \ldots\ldots\ldots, b^2}_{2^{k-1}(2n+3m)-f(k)\ times} \right) \right)$$

This language is recursive, if the function f is recursive. Then the Wätjen's proof is based on demonstration of a contradiction in such a way:

For the non-recursive f it is supposed that the language $L(\Sigma, f)$ remains recursive. This leads, however, to a contradiction in the following way: If $L(\Sigma, f)$ is a recursive language, then the words belonging to it can be effectively listed in some order. Now, choose an arbitrary $k \in N$. Then there exists a word w_k which belongs to $L(\Sigma, f)$. This word is listed after finite number of steps, and because of that we can compute the value $f(k)$ for it. So, f is computable. This is the contradiction. Consequently, the language $L(\Sigma, f)$ is non-recursive.

As we have mentioned already in [8], in the context of the sketched framework of the EG systems with agents organized into *teams* as proposed in [5] and in [16], we may imagine and suppose – in a very intuitive level – that the physical bodies of agents are the very things we want for prohibition of all of agents activities in certain extent during the rewriting process, in other words for creating teams of agents. In the case of fixed number of agents in teams [5] we recognize no principal influence of teams to the computational power of the EG systems. However, in the case of changing number of agents in teams – as proved in [16] and as we have sketched in this section – the computational power of the EG systems basically depends on the computational properties of functions which define the number of active agents in teams. Non-recursive functions appearing in this model cause the possibility of generating non-recursive languages – behaviors – using the corresponding EG systems. The non-recursiveness of the function f in the model can be interpreted as incorporation of the randomness of grouping embodied agents into teams in the model.

We conjecture that if this randomness incorporated into EG systems will be at the level expressible only through a hyper-computable functions f, then the corresponding EG systems which will use such type of functions will be – using the terminology proposed in [2] – *super-recursive*, so that they will be able to perform *hyper-computations*.

In any case, the interpretation of the result from [16] concerning the computational power of EG systems with changing number of teams in them proves that embodiment of agents significantly influences the computational power of communities of agents in comparison with the individual computational power of agents forming herds and participating on their activities. In certain sense the difference between the behavior of the agents and of the whole herds formed by them might be comprehended as a surprise required in the test of emergence formulated in [12], and the non-recursiveness (or the above hypothesized super-recursiveness) of societies of agents as an *emergent effect* of agents embodiment.

Acknowledgement

The author's research on the topic is supported by the Czech Science Foundation Grant No. 201/04/0528.

REFERENCES

[1] Brooks, R. A.: Cambrian Intelligence. The MIT Press, Cambridge, Mass., 1999.

[2] Burgin, M.—Klinger, A.: Preface – Three Aspects of Super-Recursive Algorithms and Hyper-Computation or Finding Black Swans. Theoretical Computer Science 317, 2004, pp. 1–11.

[3] Copeland, B. J.: Hypercomputation – Some Philosophical Issues. Theoretical Computer Science 317, 2004, pp. 251–267.

[4] Csuhaj-Varjú, E.—Kelemen, J.—Kelemenová, A.—Paun, Gh.: Eco-Grammar Systems – A Grammatical Framework for Lifelike Interactions. Artificial Life 3, 1997, pp. 1–28.

[5] Csuhaj-Varjú, E.—Kelemenová, A.: Team Behaviour in Eco-Grammar Systems. Theoretical Computer Science 209, 1998, pp. 213–224.

[6] Eberbach, E.—Wegner, P.: Beyond Turing machines. Bulletin of the EATCS 81, 2003, pp. 279–304.

[7] Humphrey, N.: How to Solve the Mind-Body Problem. Imprint Academic, Thorverton, 2000.

[8] Kelemen, J.: The Agent Paradigm – Foreword. Computing and Informatics, Vol. 22, 2003, pp. 513–519.

[9] Kelemen J.: May Embodiment Cause Hyper-Computation? In: Advances in Artificial Life, Proc. ECAL 05 (M. S. Capcarrére et al., eds.). Springer-Verlag, Berlin, 2005, pp. 31–36.

[10] Minsky, M.: The Society of Mind. Simon and Schuster, New York, 1986.

[11] Parker, L.: Current Research in Multirobot Systems. Artificial Life and Robotics 7, 2003, pp. 1–5.

[12] Ronald, E. M. A.—Sipper, M.—Capcarréré, M. S.: Design, Observation, Surprise! A Test of Emergence. Artificial Life 5, 1999, pp. 225–239.

[13] Rozenberg, G.—Salomaa, A.: The Mathematical Theory of L-Systems. Academic Press, New York, 1980.

[14] Scheutz, M.: Computationalism – The Next Generation. In: Computationalism (M. Scheutz, Ed.). The MIT Press, Cambridge, Mass., 2002, pp. 1–21.

[15] Sloman, A.: The Irrelevance of Turing Machines to Artificial Intelligence. In: Computationalism (M. Scheutz, Ed.). The MIT Press, Cambridge, Mass., 2002, pp. 87–127.

[16] Wätjen, D.: Function-Dependent Teams in Ecogrammar Systems. Theoretical Computer Science 306, 2003, pp. 39–53.

[17] Wegner, P.: Why Interaction Is More Powerful than Algorithms. Communications of the ACM 40, 1997, No. 5, pp. 81–91.

[18] Wegner, P.—Goldin, D.: Computing Beyond Turing Machines. Communications of the ACM 46, 2003, No. 4, pp. 100–102.

Jozef Kelemen received his degrees in mathematics at the Comenius University, Bratislava, Slovakia, in theoretical cybernetics at the Academy of Sciences, Moscow, Russia, and in computing technology at the Slovak Technical University, Bratislava, Slovakia. In the past, he was associated (in the positions of associate or full professor) with the Comenius University and University of Economics, Bratislava, Slovakia, and with Lorand Eotvos University, Budapest, and Istvan Szechenyi University of Technology, Gyor, Hungary, among others. Now, he is a full professor of computer science and the head of the Institute of Computer Science at the Silesian University at Opava, Czech Republic, and a research fellow of the IT company Gratex International. His professional interests include some branches of theoretical computer science, artificial intelligence, artificial life, and cognitive science. He is a member of editorial boards of the journals Computing and Informatics, Experimental and Theoretical Artificial Intelligence, Grammars, and Neural Network World, and of several international program committees of symposia and conferences, a member of the American Association for Artificial Intelligence (AAAI), and the honorary member of the Hungarian Fuzzy Association.