Computing and Informatics, Vol. 22, 2003, 623–637

A NOTE ON EMERGENCE IN MULTI-AGENT STRING PROCESSING SYSTEMS

Rudolf Freund

Faculty of Computer Science
Vienna University of Technology
Favoritenstrasse 9, A-1040 Wien, Austria
e-mail: rudi@emcc.at

Jozef Kelemen[*]

Institute of Computer Science, Silesian University
Bezručovo nám. 13, 746 01 Opava, Czech Republic
and
Gratex International, a. s.
Bratislava, Slovakia
e-mail: kelemen@fpf.slu.cz

Gheorghe Păun

Institute of Mathematics, Romanian Academy
PO Box 1-764, 70700 Bucureşti, Romania
and
Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tarraco 1, 43005 Tarragona, Spain
e-mail: gp@astor.urv.es

Revised manuscript received 11 November 2003

[*] Author's research on the subject is partially supported by the Czech Science Foundation Grant No. 201/04/0528.

Abstract. We propose a way to define (and, in a certain extent, even to measure) the phenomenon of emergence which appears in a complex system of interacting agents whose global behaviour can be described by a language and whose components (agents) can also be associated with grammars and languages. The basic idea is to identify the "linear composition of behaviours" with "closure under basic operations", such as the AFL (Abstract Families of Languages) operations, which are standard in the theory of formal languages.

Keywords: Grammar systems, multi-agent systems, emergence, abstract families of languages

1 INTRODUCTION

The experiences with different fields of computing as well as with some non-computational systems like behaviour-based (collective) robotics [1, 2, 3, 4], artificial life [23], multi-agent systems [14], artificial economies [13], etc. in the last decade of the 20th century resulted in considering a new possibility of how to set up multi-component systems from autonomous components usually called *agents*. In the fields mentioned above usually the larger or the more complex systems (*multi-agent systems*) evolve incrementally, usually by adding new components to the existing ones (and/or by deleting some others). In many cases this is the only possible way to change the architecture of the systems. The communication between these components is usually very opportunistic and guided by the desired global behaviour of the resulting systems. Consequently, the modularity of such systems – the so-called *post-modularity* [34] – is principially different from those of the previous ones. The behaviour of such systems *emerges* from massive interactions between the components of the system and their environment. In this paper we propose a way to define and – in a certain extent – to even measure the phenomenon of emergence which appears in a complex system whose global behaviour can be described by a language and whose components can also be associated with languages. The basic idea is to identify the "linear composition of behaviours" with "closure under basic operations", such as the AFL operations (which are standard in language theory).

2 GRAMMAR SYSTEMS – EMERGENCE IN A NATURAL FRAMEWORK

The aim of this section is to informally describe some of the main ideas of grammar systems theory emphasizing the fact that the idea of emergence is central in this area. Later, making use of the deeply developed theory of formal languages, where grammar systems theory is embedded, we will propose a rather operational way to approach the notion of emergence.

In "classic" formal language theory, grammars (and automata) are used separately, *one* grammar generates *one* language (*one* automaton recognizes *one* language). However, in modern computer science, such notions as distribution, co-operation, parallelism, communication, synchronization have become more and more popular and important. Computer networks, parallel computing, distributed data bases, etc., are practical counterparts (sources) of these notions. It is highly expectable that also the human brain is a decentralized system working in a distributed manner, if we realize the number of components the brain uses simultaneously at each moment of its work; see [25], for instance, for more details on a hypothesis about that.

In (theoretical) computer science there are several approaches to these ideas, in general using algebraic models or automata models. A grammatical model of distributed/parallel work of several agents has been developed, mainly in the last decade, under the name of *grammar systems theory.* This is already a well elaborated branch of formal language theory. The monograph [6] contains the results known at the middle of 1992. Many subsequent developments appear in the literature; here we only mention the collective volumes [27, 28, 29], as well as the proceedings of the series of workshops about grammar systems held every second year after 1994.

In short, a grammar system is a *set* of grammars, working together, according to a specified protocol, for producing *one* language. The crucial element here is the protocol of co-operation. The aim of considering such a compound generative machinery can be many-sided: to model a real phenomenon, to increase the (generative) power of the components, to decrease the (descriptional) complexity. The co-operation protocol has to deal with all these aspects. In some sense, the theory of grammar systems is the theory of co-operation protocols; the focus does not lie on the generative capacity, but on the functioning of the systems, and on its influence on the generative capacity and on other specific properties. In other words, the main interest lies on the global behaviour of the system, in comparison and in relation with the behaviour of the components. As expected, the emergence phenomenon is crucial in this framework, what the system can do is always much more than "the sum" of the components output. We will give some details below.

Up to now, two basic classes of grammar systems have mainly been investigated, the *sequential* ones and the *parallel* ones.

In a *co-operating distributed* (CD) grammar system (introduced in [5]), several grammars have a common sentential form. Initially, this is a common axiom. At each moment, one grammar is active and rewrites the string, whereas the others are inactive. The conditions under which a component can become active, when it will become inactive again and leave the sentential form to the other components – these aspects are very important (also for the power of the system) and they are specified by the co-operation protocol. The language of terminal strings generated in this way is the language generated by the system. The following basic stop conditions are given: each component, when active, has to work for "exactly k steps", "at least k steps", "at most k steps", "any number of steps", or "the maximal number of steps" (a step means the application of a rewriting rule). Many other starting and stopping conditions were considered in the literature, sometimes also an external control was imposed, specifying the order of components to be enabled, and ways to increase collaboration between components were also taken into account.

In *parallel communicating* (PC) grammar systems (introduced in [30]), each component has its own sentential form. In each time unit (a common clock divides the time in units, in a uniform way for all components) each component uses a rule, rewriting the associated sentential form. Up to now, we have separate grammars working separately; what transforms the construct into a *system* is the possibility of communicating. Special (query) symbols are provided, pointing to components of the system. When a component i introduces the query symbol Q_j, then the current sentential form of the component j will be sent to the component i, replacing the occurrence of Q_j. One component is distinguished as the *master*, and the language generated by it, alone or involving communications, is the language generated by the system. Several variants can be considered, depending on the shape of the communication graph, on the action a component has to perform after communicating, and so on.

An interesting particular case of a CD grammar system is that of a *colony* of grammars (introduced in [21]), where the components are considered as simple as possible, generating only finite sets of strings. In the other direction, a complex combination of both CD and PC grammar systems is found in an *eco-grammar system* (introduced in [8]), where several agents, described by strings, evolve according to and act on a common environment by means of rewriting rules; these rules are chosen according to the state of the agent; the environment evolves independently from the states of the agents, whereas the agents may also interact with each other. A generalization of PC grammar systems (especially of those where the communication is done by command, like in [9]) was also proposed in [10].

Using the concept of colonies we now will illustrate how more complicated phenomena may emerge from the simplest ones.

Let us first consider the notion of the *emergent computation* explained in [15] as follows. An *emergent computation* is supposed to consist of:

1. a collection of agents, each following explicit *instructions*;
2. *interactions* between the agents (according to the instructions), which form implicit global patterns at the macroscopic level, i.e., *epiphenomena*;
3. a natural *interpretation* of the epiphenomena as computation.

It is clear that all components in a colony of grammars follow explicit *instructions.* These instructions are formulated in the form of rewriting rules and the process of their use is formulated explicitly in the definition of the derivation step. *Interactions* between the components are limited by the strategy of co-opration and their behaviour "at the macroscopic level", so observing the behaviour of the whole colony, instead of the individual behaviours of its components, produces – as an epiphenomenon – an infinite language, which is naturally *interpretable* as a computation: it can be generated/accepted by the corresponding type of an abstract computing device (if it is a context-free language, which is the typical case for colonies, then by the corresponding pushdown automaton).

E. M. A. Ronald et al. [31] have formulated a *test of emergence* trying to offer an operational definition of emergence for artificial life experiments. The requirements put onto a system in which the phenomenon of emergence appears are as follows:

Design. The designer designs the systems by describing *local* interactions between components in a language L_1.

Observations. The observer describes global behaviours of the running system using a language L_2.

Surprise. The language of design L_1 and the language of observation L_2 are distinct, and the causal link between the elementary interactions programmed in L_1 and the observations observed in L_2 are non-obvious.

Let us analyze the *design* in more details. Imagine a designer (or more designers) "programming" (constructively defining) the particular simple grammars, say R_1 and R_2, for generating some required finite numbers of strings, say $\{aB\}$ and $\{Ab, b\}$ (for instance, we may take the very simple regular grammars $R_1 = (\{A\}, \{a, B\}, \{A \to aB\}, A)$ and $R_2 = (\{B\}, \{A, b\}, \{B \to Ab, B \to b\}, B)$). The designer is satisfied because the work of the constructed modules fits the given requirements. Using both of these generative devices in isolation, they generate the simple sum of the two behaviours, the *finite* language $\{aB, Ab, b\}$. No surprise appears during the observation of such a kind of system created from the (just described) *isolated* modules.

Now, imagine that an observer puts the grammars together to rewrite a shared sentential form starting with a non-terminal symbol from one of the simple grammars (in our example, we start with the non-terminal symbol A from R_1). What will be *observed* now? The global behaviour of the whole system will be different from the simple finite "sum" $\{aB, Ab, b\}$ of the behaviours of the indvidual simple grammars. The observed behaviour will be an *infinite* language $\{a^n b^n \mid n \geq 1\}$. However, the design of the two components remains unchanged! So, while the language of the *designer* has been concerned with *finite* behaviours, the language of the *observer* of the system created in a rather simple way from the designed components will be concerned with *infinite* behaviours (languages).

The impression of a *surprise* arises by realizing the gap between finite and infinite behaviour just demonstrated above for the two systems set up from the same components but working in different environments (starting with different starting strings). Theoretically, some more surprising facts have been proved for colonies. As we have already mentioned, colonies – devices set up from regular grammars generating finite languages – define the whole family of context-free languages, and if they work with some simple and very "natural" additional restrictions, their generative capacity can even be enlarged to some context-sensitive languages, too. Moreover, in the case of the finite languages some problems seem to be highly actual, for instance the complexity of defining such languages, and some of them are completely trivial, like problems concerning decidability, computability, etc. Some questions become to be meaningful only in the case of infinite languages. In other words, there are two (not completely distinct, of course) languages for talking (and thinking) about finite languages and devices generating them, and, in contrast, for infinite languages.

Now, we could state that colonies satisfy the test of emergence as proposed in [31], and that they offer a quite simple and theoretically well grounded example of how the test can be applied. In such a case, moreover, we can demonstrate the appearance of the phenomenon of emergence in theoretically (mathematically) well described formal systems, too, not only in experimental situations as discussed in [31]. More about this can be found in [20]. In more technical details, the example mentioned above can be found in [20].

In all these classes of grammar systems the phenomenon of emergence is fundamental, and the main reason of considering them is the fact that, *in most cases, the language generated* (sometimes, accepted) *by the system is essentially more complex than the languages of the components.* Just to have an idea about what this means, we recall a series of results, in an informal manner (an elementary knowledge of basic elements of formal language theory, especially about the Chomsky hierarchy, is assumed):

1. CD grammar systems with regular components generate only regular languages, in all modes of co-operation, but systems with context-free components can generate non-context-free languages in the modes "at least k steps", "exactly k steps", "maximal number of steps" ([5, 6]).
2. The colonies with sequential rewriting characterize the family of context-free languages ([7]), but colonies with parallel rewriting or with an enhanced co-operation of components can generate non-context-free languages ([12, 22, 26]).
3. PC grammar systems with regular components can generate non-context-free languages ([6, 30]), while systems with context-free components characterize the recursively enumerable languages ([11, 24]).

3 ABSTRACT FAMILIES OF LANGUAGES: A POSSIBLE THEORETICAL FRAMEWORK

The difference between the level of components (finite in the case of colonies, regular in the case of PC grammar systems, context-free in the case of CD and PC grammar systems) and the level of the system output (context-free or more in the case of colonies, non-context-free in the case of CD grammar systems, non-context-free or even recursively enumerable in the case of PC grammar systems) is striking, and this is exactly the effect of emergence. It is also clear that we can almost directly apply the *test of emergence*, as proposed in [31]; see [20] for further discussions about this topic, mainly about the case of colonies.

However, what we want to have is a more operational concept of a “test of emergence”, a more formal definition and, if possible, a way to quantify it. The main difficulty lies in the fact that the very concept of emergence is somewhat … *emergent*, mainly an intuition, a feeling (the feeling of a surprise, see [31]). The test of emergency proposed in [31] tries to capture the meaning of the concept in a more precise manner, but still a lot remains imprecise, at least from a mathematical point of view. Of course, we cannot hope to have a precise definition of a general notion as that of emergence able to cover the concept in its whole generality, hence, we here will try to approach it only for systems having to do with string processing. The case study which we will discuss will be that of grammar systems, which perfectly fits in this category.

At a more technical level, the difficulty is to transfer to our domain and to formally define the meaning of the intuitive statement that “emergence appears when the system behaviour is [much] more than the *linear sum* of the components behaviour”.

Our basic proposal is to interpret the “linear sum” in terms of “nice operations on languages”; then, the linear closure of a set (of behaviours) will mean the closure of a set of languages under given “nice” operations.

Fortunately, in formal language theory we have several possibilities to choose “nice” operations, namely those used in the abstract theory of languages (in short, AFL), a branch of formal languages much developed in the sixties and the seventies. The monograph [16] is just a milestone of that area; further references can be found in [32].

The basic notions of AFL theory are those of *trio, semi-AFL*, and *AFL*: A *trio* is a family of languages which is closed under non-erasing morphisms, inverse morphisms, and intersection with regular languages. A *semi-AFL* is a trio closed also under union. An *AFL* is a semi-AFL also closed under concatenation and Kleene $+$. Thus, in sum, an AFL is a family of languages which is closed under union, concatenation, Kleene $+$, non-erasing morphisms, intersection with regular languages, and inverse morphisms. (These operations are not independent of each other, but they are basic in formal language theory and, in this combination, considered in the AFL theory; further discussions about this topic can be found in [16].) A trio, semi-AFL, or AFL which is closed under arbitrary morphisms and under Kleene $*$ is said to be *full*.

As most important examples, we mention that the families of regular languages (REG), of context-free languages (CF), of context-sensitive languages (CS), and of recursively enumerable languages (RE) are AFLs, while the family of linear languages (LIN) is a semi-AFL which not an AFL (it is not closed under concatenation and Kleene closure). All these families are full, with the exception of CS, which is not closed under erasing morphisms.

It is also very important for our purposes to note that REG is the smallest trio, semi-AFL, and AFL. Actually, a much more impressive statement holds: *If F is any full trio such that there is a language in F which is non-empty, then $REG \subseteq F$* (Theorem 3.1.1, [16]).

Now, starting from a given family F of languages (in particular, it can be finite, or only consisting of only one language), we can look for the smallest trio, semi-AFL, and AFL which contains the languages from F ("smallest" here is meant in the sense of the inclusion relation), and we denote the resulting families of languages by $trio(F)$, $sAFL(F)$, and $AFL(F)$, respectively.

Intuitively, such families consist of all languages from F, plus all languages which can be obtained from the languages of F by applying to them the operations corresponding to the type of family we look for, putting together the languages of F and all those obtained so far, and then iterating this procedure an arbitrary number of times. Thus, it is clear that $F \subseteq trio(F) \subseteq sAFL(F) \subseteq AFL(F)$. Of course, depending on the initial family F, some of these inclusions can be proper, whereas others can be equalities.

A family F of languages such that $F = trio(\{L\})$ is said to be a *principal trio*, while L is said to be a *generator* of F. The notion directly extends to principal semi-AFLs and principal AFLs. For our purposes, it is important to note that *every principal semi-AFL or AFL has an infinite number of generators* (Exercise 5.1.1, [16]).

All the families REG, LIN, CF, CS, and RE are principal semi-AFLs (and AFLs in the case of REG, CF, CS, and RE). As we have remarked above, any non-empty language is a generator of the full AFL REG. A generator of LIN is, for instance, the language $\{w\ mi(w) \mid w \in \{a,b\}^*\}$, where $mi(w)$ denotes the mirror image of the string w. Well-known generators of CF are the Dyck languages and generators of RE are the twin-shuffle languages. We do not enter into further details here, yet we return to the goal of our investigation, a framework for defining emergence.

Convention: All languages we deal with in this paper are recursively enumerable, so we do not consider languages which are not Turing enumerable.

4 DEFINING EMERGENCE – THE CASE OF GRAMMAR SYSTEMS

In [18] something what is *emergent* is explained as "... a product of coupled, context-dependent interactions. Technically, these interactions and the resulting system are *nonlinear*: The behaviour of the overall system *cannot* be obtained by *summing* the behaviours of its constituent parts... However, we *can* reduce the behaviour of the whole to the lawful behaviour of its parts, *if* we take nonlinear interactions into account." In what follows we will propose a way how this formulation can be translated into a more sophisticated formal framework.

Consider a system Σ, composed of several subsystems $\sigma_1, \ldots, \sigma_n$. Assume that with the system itself and with its subsystems we can, in a natural way, associate some languages $L(\Sigma), L(\sigma_1), \ldots, L(\sigma_n)$, which describe the behaviour (the type, the competence) of the system and of each of its subsystems.

- If $L(\Sigma) \in trio(\{L(\sigma_1), \ldots, L(\sigma_n)\})$, then we say that the system displays *no emergence.*

- If $L(\Sigma) \notin trio(\{L(\sigma_1), \ldots, L(\sigma_n)\})$, then we say that the system displays *a trio emergence.*
- If $L(\Sigma) \notin sAFL(\{L(\sigma_1), \ldots, L(\sigma_n)\})$, then we say that the system displays *a semi-AFL emergence.*
- If $L(\Sigma) \notin AFL(\{L(\sigma_1), \ldots, L(\sigma_n)\})$, then we say that the system displays *an AFL emergence.*

A further level can also be considered: let $AFL_\cap(F)$ denote the smallest AFL which contains F and is closed under intersection.

- If $L(\Sigma) \notin AFL_\cap(\{L(\sigma_1), \ldots, L(\sigma_n)\})$, then we say that the system displays *a strong AFL emergence.*

Let us see how these precise notions of emergence apply to grammar systems of various types.

If the component languages are regular, then the smallest trio, semi-AFL, AFL containing them is REG. If the component languages are context-free, then the smallest trio, semi-AFL, AFL containing them is CF. This means that the colonies have an AFL emergence, which is not the case for CD grammar systems with regular components, but is true again for CD grammar systems with context-free components, and for PC grammar systems with both regular or context-free components.

However, the results we have mentioned at the end of Section 2 about the power of grammar systems in some sense show more than the previous estimations. For instance, colonies with parallel rewriting not only generate languages outside REG, the smallest AFL containing the finite languages, but also outside CF (implicitly, outside the semi-AFL LIN). Similarly, PC grammar systems with regular components have the behaviour outside CF, while systems with context-free components jump directly to the full AFL of all recursively enumerable languages.

Moreover, because REG is closed under intersection, it follows that always when we have finite or regular component languages, trio emergence directly means strong AFL emergence.

These observations suggest that the previous scale for assessing the emergent behaviour of a system is operational enough, but not refined enough. A solution is to take a given hierarchy of trios, semi-AFLs, or AFLs as reference, as we have done above with the basic families in the Chomsky hierarchy. Several other families can be placed in between these five families of languages. At the limit, we can consider *infinite* hierarchies of families with given closure properties.

Consider such an infinite hierarchy of, for instance, AFLs, $\mathcal{H} : REG = F_1 \subset F_2 \subset \ldots \subset F_i \subset F_{i+1} \subset \ldots \subseteq RE$ (remember that we only work with Turing enumerable languages). Then, if $L(\sigma_j) \in F_m, 1 \leq j \leq n$, and $L(\Sigma) \in F_{m+t} - F_{m+t-1}$, for some $t \geq 1$, then we will say that Σ is t-emergent with respect to the hierarchy $\mathcal{H}$. Of course, in this framework we can also have an infinite emergence: if $L(\sigma_j) \in F_m$ and $L(\Sigma)$ belongs to no $F_r, r \geq m$, then Σ is ∞-emergent, a property of a clear interest.

In this way, we can have a precise *numerical* estimation of the emergent behaviour of Σ.

For instance, if we take the (Chomsky) hierarchy $\mathcal{H}_{Ch} : REG \subset LIN \subset CF \subset CS \subset RE$ (of semi-AFLs), we know that sequential colonies are 2-emergent, while parallel colonies are 3-emergent, CD grammar systems with context-free rules are 1-emergent, PC grammar systems with regular components are 3-emergent, and PC grammar systems with context-free components are 2-emergent, always with respect to the chosen hierarchy, i.e., the Chomsky hierarchy $\mathcal{H}_{Ch}$.

It is visible that this refined notion of emergence is rather suggestive – and it can even be more suggestive for a more detailed hierarchy of families.

Just as an illustration, let us consider an infinite hierarchy of trios of context-free languages, the one defined by *the index* of context-free grammars.

For a derivation

$$\delta : S = w_0 \Longrightarrow w_1 \Longrightarrow \ldots \Longrightarrow w_m = z \in T^*$$

in a context-free grammar $G = (N, T, P, S)$ (the notion can be extended to any class of grammars using non-terminal symbols) we define

$$ind(\delta) = \max\{|w_i|_N \mid 0 \le i \le m\},$$

where $|x|_N$ is the number of occurrences of non-terminal symbols from N in the string $x \in (N \cup T)^*$; the maximal number of occurrences of non-terminal symbols from N in all the sentential forms is the index of the derivation δ. Then, if $\Delta(z, G)$ is the set of derivations of $z \in T^*$ with respect to the grammar G, the index of z with respect to G is defined by

$$ind(z, G) = \max\{ind(\delta) \mid \delta \in \Delta(z, G)\}.$$

The index of the grammar G is

$$ind(G) = \sup\{ind(z, G) \mid z \in L(G)\},$$

and for a language $L \in CF$ the index with respect to the class of context-free grammars is

$$ind_{CF}(L) = \inf\{ind(G) \mid L = L(G)\}.$$

Now let us denote the family of languages $\{L \in CF \mid ind_{CF}(L) \le i\}$, $i \ge 1$, by CF_i. It is known (see, e.g., [17]) that all the inclusions $CF_i \subset CF_{i+1}$, $i \ge 1$, are proper and that all these families are full trios. Moreover, it is shown in [33] that there are context-free languages of an infinite index – an example is the Dyck language over $\{a, b\}$. Consequently, we have the following hierarchy of trios:

$$LIN = CF_1 \subset CF_2 \subset \ldots \subset CF_i \subset CF_{i+1} \subset \ldots \subset CF.$$

Now, let us consider this hierarchy as the reference for emergence:

At first sight, colonies offer us the pleasant result of having an infinite emergence: the components of a colony are regular grammars, hence they correspond to index 1, while the family of languages generated by colonies equals CF, hence also languages of an infinite index are reached. The surprise is only partially justified, because of the apparent incongruence between the definition of a colony and the definition of the index: in a colony, the terminal symbols of one component can be rewritten by another component, hence they become nonterminals for the second component (hence for the colony as a whole), while in the definition of the index there is an essential distinction between terminals and nonterminals – and only the latter are counted. Still, the behaviour of colonies can be considered as surprising even in that context.

Then, a natural case to be considered is that of PC grammar systems, where the separation of terminals and non-terminals is rigid and systems with regular components (index 1) are able to generate languages which are not regular. Here we are only interested in context-free languages.

Which is the maximal index of a context-free language which can be generated by a PC grammar system with regular components? Actually, we here have four problems, depending on the type of systems we consider: centralized or non-centralized, returning or non-returning; all these problems remain *open* here.

We can only prove that 1-emergence appears, at least for returning PC grammar systems, centralized or non-centralized, and this can be shown with the following system:

$$\begin{aligned}
\Gamma &= (N, K, T, (P_1, S_2), (P_2, S_2)),\\
N &= \{S_1, S_2\},\\
K &= \{Q_1, Q_2\},\\
T &= \{a, b\},\\
P_1 &= \{S_2 \to a,\ S_2 \to aS_1,\ S_1 \to aS_1,\ S_1 \to aQ_2\},\\
P_2 &= \{S_2 \to bS_2\}.
\end{aligned}$$

Note that the axiom of both components is S_2 and that the system is centralized and regular in the restricted sense (the rules are not right-linear, but regular). Here is a typical derivation in Γ, in the returning mode:

$$\begin{aligned}
(S_2, S_2) &\Longrightarrow (aS_1, bS_2) \Longrightarrow^* (a^nS_1, b^nS_2) \qquad \text{for some } n \geq 1\\
&\Longrightarrow (a^{n+1}Q_2, b^{n+1}S_2) \Longrightarrow (a^{n+1}b^{n+1}S_2, S_2)\\
&\Longrightarrow (a^{n+1}b^{n+1}aS_1, bS_2) \Longrightarrow^* (a^{n+1}b^{n+1}a^mS_1, b^mS_2) \qquad \text{for some } m \geq 1\\
&\Longrightarrow (a^{n+1}b^{n+1}a^{m+1}Q_2, b^{m+1}S_2) \Longrightarrow^* (a^{n+1}b^{n+1}a^{m+1}b^{m+1}S_2, S_2)\\
&\Longrightarrow (a^{n+1}b^{n+1}a^{m+1}b^{m+1}a, bS_2).
\end{aligned}$$

Consequently, the language generated by Γ in the returning mode is

$$L_r(\Gamma) = \{a^nb^n \mid n \geq 2\}^*\{a\}.$$

This language is not linear, but it can be generated by a context-free grammar of index 2. Thus, we have proved 1-emergence.

We conclude this section with a technical observation: if one of the languages $L(\sigma_i)$ is a generator of the full semi-AFL LIN (remember that there are infinitely many such generators), then the smallest full semi-AFL which contains this language and is closed under intersection is RE. Indeed, each recursively enumerable language is the morphic image of the intersection of two linear languages. Consequently, in such a case the system cannot display any strong semi-AFL (or AFL) emergence, we already have $sAFL(\{L(\sigma_1), \dots, L(\sigma_n)\}) = RE$.

5 FINAL REMARKS

We have introduced a formal framework where the intuitive (we may also say "mysterious") notion of emergence can be formulated (and "measured") precisely for complex systems whose behaviour can be described by string languages. The main idea is to consider hierarchies of language families with 'nice' closure properties (as a language approximation of what a 'linear composition' could mean for the behaviour of the components of the system) and to evaluate the jump in the hierarchy from the level of languages of the components to the level of the language describing the behaviour of the whole system. The idea has been illustrated with the case of grammar systems (CD grammar systems, PC grammar systems, colonies), for the Chomsky hierarchy of languages and for the infinite hierarchy of families of context-free languages defined by the index of context-free grammars. Our study has a preliminary character, as further investigations should be done both at the technical level (in particular, an open problem which we feel interesting was formulated with respect to the emergent behaviour of PC grammar systems with regular components) and in what concerns case studies of "real life" complex systems, whose emergent behaviour could be evaluated in this framework.

REFERENCES

[1] Arkin, R. C. Ed.: Robot Colonies. Kluwer, Boston, Mass., 1997.

[2] Arkin, R. C.: Behaviour-Based Robotics. The MIT Press, Cambridge, Mass., 1998.

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

[4] Connell, J. H.: Minimalist Mobile Robotics – a Colony Architecture for a Mobile Robot. Academic Press, New York, 1990.

[5] Csuhaj-Varjú, E.—Dassow, J.: On Co-Operating Distributed Grammar Systems. J. Inform. Process. Cybern., EIK, Vol. 26, 1990, pp. 49–63.

[6] Csuhaj-Varju, E.—Dassow, J.—Kelemen, J.—Păun, Gh.: Grammar Systems. A Grammatical Approach to Distribution and Co-operation. Gordon and Breach, London, 1994.

[7] Csuhaj-Varjú, E.—Kelemenová, A.: Languages of Colonies. Theoretical Computer Science, Vol. 134, 1994, pp. 119–130.

[8] Csuhaj-Varjú, E.—Kelemen, J.—Kelemenová, A.—Păun, Gh.: Eco-Grammar Systems: A Grammatical Framework for Studying Life-Like Interactions. Artificial Life, Vol. 3, 1997, pp. 1–28.

[9] Csuhaj-Varjú, E.—Kelemen, J.—Păun, Gh.: Grammar Systems with WAVE-Like Communication. Computers and AI, Vol. 15, 1996, pp. 419–436.

[10] Csuhaj-Varjú, E.—Salomaa, A.: Networks of Parallel Language Processors. In: [28], pp. 299–318.

[11] Csuhaj-Varjú, E.—Vaszil, G.: On the Computational Completeness of Context-Free PC Grammar Systems. Theoretical Computer Science, Vol. 215, 1999, pp. 348–358.

[12] Dassow, J.—Kelemen, J.—Păun, Gh.: On Parallelism in Colonies. Cybernetics and Systems, Vol. 24, 1993, pp. 37–49.

[13] Epstein, J. M.—Axtell, R.: Growing Artificial Societies – Social Science from the Bottom Up. The MIT Press, Cambridge, Mass., 1996.

[14] Ferber, J.: Multi-Agent Systems – an Introduction to Distributed Artificial Intelligence. Addison-Wesley, Harlow, 1999.

[15] Forrest, S.: Emergent Computation – Self-Organizing, Collective, and Co-Operative Phenomena in Natural and Artificial Computing Networks. In: Emergent Computation (S. Forrest, Ed.). The MIT Press, Cambridge, Mass., 1991, pp. 1–11.

[16] Ginsburg, S.: Algebraic and Automata-Theoretic Properties of Formal Languages. North-Holland, Amsterdam, 1975.

[17] Gruska, J.: Descriptional Complexity of Context-Free Languages. Proc. MFCS Symp., High Tatras, Czechoslovakia, 1973, pp. 71–83.

[18] Holland, J. H.: Emergence – from Chaos to Order. Addison-Wesley, Reading, Mass., 1998.

[19] Kelemen, J.: On Post-Modularity and Emergence from Grammar-Theoretic Point of View. In: Quo Vadis Computational Intelligence? (P. Sinčák, J. Vaščák, Eds.), Physica-Verlag, Heidelberg, 2000, pp. 342–352.

[20] Kelemen, J.: From Statistics to Emergence – Exercises in Systems Modularity. In: Multi-Agent Systems and Applications (M. Luck et al., Eds.), Springer-Verlag, Berlin, 2001, pp. 281–300.

[21] Kelemen, J.—Kelemenová, A.: A Grammar-Theoretic Treatment of Multiagent Systems. Cybernetics and Systems, Vol. 23, 1992, pp. 210–218.

[22] Kelemenová, A.: Timing in Colonies. In: [29], pp. 136–143.

[23] Langton, Ch. Ed.: Artificial Life – an Overview. The MIT Press, Cambridge, Mass., 1995.

[24] Mandache, N.: On the Computational Power of Context-Free PCGS. Theoretical Computer Sci., Vol. 237, 2000, pp. 135–148.

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

[26] Păun, Gh.: On the Generative Capacity of Colonies. Kybernetika, Vol. 31, 1995, pp. 83–97.

[27] Păun, Gh. Ed.: Artificial Life. Grammatical Models. Black Sea Univ. Press, Bucureşti, 1995.

[28] Păun, Gh.—Salomaa, A. Eds.: New Trends in Formal Languages: Control, Co-Operation, Combinatorics. Springer-Verlag, Berlin, 1997.

[29] Păun, Gh.—Salomaa, A. Eds.: Grammatical Models of Multi-Agent Systems. Gordon and Breach, London, 1998.

[30] Păun, Gh.—Sântean, L.: Parallel Communicating Grammar Systems: the Regular Case. Ann. Univ. Buc., Matem.-Inform. Series, Vol. 38, 1989, No. 2, pp. 55–63.

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

[32] Rozenberg, G.—Salomaa, A. Eds.: Handbook of Formal Languages, 3 volumes, Springer-Verlag, Berlin, 1997.

[33] Salomaa, A.: On the Index of Context-Free Grammars and Languages. Inform. Control, Vol. 14, 1969, pp. 474–477.

[34] Stein, L. A.: Post-Modular Systems – Architectural Principles for Cognitive Robotics. Cybernetics and Systems, Vol. 28, 1997, pp. 471–487.

Rudolf Freund received his master and doctor degree in computer science from the Vienna University of Technology, Austria, in 1980 and 1982, respectively. In 1986, he received his master degree in mathematics and physics from the University Vienna, Austria. In 1988 he joined the Vienna University of Technology in Austria, where he became an Associate Professor in September 1995. His research interests include array and graph grammars, regulated rewriting, infinite words, syntactic pattern recognition, neural networks and their applications, and especially models and systems for biological computing. In these fields he is author or co-author of more than ninety scientific papers.

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.

Gheorghe Păun has graduated from the Faculty of Mathematics, University of Bucharest, in 1974 and received his Ph.D. in Mathematics (specialization: Computer Science) from the same university in 1977. He held a research position at the University of Bucharest, and from 1990 he is at the Institute of Mathematics of the Romanian Academy, where he is currently a senior researcher. From 2001 he also has a Ramon y Cajal research professor position in Spain, first working in Rovira i Virgili University, Tarragona, and currently in Sevilla University. He visited numerous universities in Europe, Asia, and North America, with frequent and/or longer stays in Turku (Finland), Leiden (The Netherlands), Magdeburg (Germany, including the Alexander von Humboldt Fellowship in 1992–93), Tarragona (Spain), London, Ontario (Canada), Rome (Italy), Tokyo (Japan), Warsaw (Poland), and Vienna (Austria).

His main research areas are formal language theory and its applications, computational linguistics, DNA computing, and membrane computing (a research area initiated by him, belonging to natural computing). He has published over 400 research papers (collaborating with many researchers worldwide), has lectured at over 100 universities, and gave numerous invited talks at recognized international conferences. He has published eleven books in mathematics and computer science, has edited over twenty five collective volumes, and also published many popular science books and books on recreational mathematics (games).

He is on the editorial boards of fourteen international journals in mathematics, computer science, and linguistics, and was/is involved in the program/steering/organizing commitees for many recognized conferences and workshops.

In 1997 he was elected a member of the Romanian Academy.