in: Proceedings of the Sixth International Conference on Artificial Life
Christoph Adami, Richard K. Belew, Hiroaki Kitano, and Charles E. Taylor, Editors,
pp. 95-103, MIT Press, Cambridge MA, 1998

Mesoscopic Analysis of Self-Evolution in an Artificial Chemistry

Peter Dittrich and **Jens Ziegler** and **Wolfgang Banzhaf**
University of Dortmund, Dept. of Computer Science, D-44221 Dortmund, Germany
http://ls11-www.informatik.uni-dortmund.de
dittrich || ziegler || banzhaf@LS11.informatik.uni-dortmund.de

Abstract

In an algorithmic artificial chemistry the objects (molecules) are data and the interactions (reactions) among them are defined by an algorithm. The same object can appear in two forms: (1) as a machine (operator) or (2) as data (operand). Thus, the same object can, on the one hand, process other objects or, on the other hand, it can be processed. This dualism enables to implicitly define a constructive artificial chemistry which exhibits quite complex behavior. Remarkably, even evolutionary behavior emerged in our experiments, without defining any explicit variation operators or fitness-function. In addition to microscopic methods (e.g., monitoring the actions of single molecules) and macroscopic measures (e.g., diversity or complexity) we developed a stepwise mesoscopic analysis method based on classification and dynamic clustering. Knowledge about the system is accumulated by an iterative process in which measuring tools (classificators) extract information which in turn is used to create new classificators.

Keywords:

self-organization, evolution, computational chemistry, constructive dynamical systems, algorithmic chemistry, chemical computing, cluster analysis, binary string system, automata reaction

Motivation

The construction of artificial life systems with complex behaviors is surprisingly easy (Ray, 1992). Analysis is often much harder, especially in population-based systems with a huge number of diverse and interacting individuals. In this case the reduction of experimental data is necessary. Measuring macroscopic parameters only (i.e. temperature, diversity or complexity) allows to see that something is happening, but does not provide insight into how or what. Microscopic tracing of an experiment and understanding every single step is possible in principal. But with growing computational resources, this becomes more and more difficult, probably impossible, because time complexity for the generation of n microscopic events (e.g., $O(n)$) is lower than time complexity needed for analysis (e.g., $O(n \log(n)$ for sorting the population). Therefore, methods for analysis in between macroscopic and microscopic methods are required.

In this paper we present a stepwise mesoscopic analysis method for artificial chemistries, which might be applied to other population-based systems as well. In this mesoscopic analysis the population is decomposed into groups. The decomposition is based on so called **classifiers**[1]. A classifier represents a property p of an object. A first-order classifier is simply a function $C : S \longrightarrow [0,1]$. The value $C(s) = p$ represents the strength of a property C of a string s and is used for a separation of the population into partial quantities with a cluster algorithm. Data produced by the classifiers as well as the collection of classifiers themselves represent knowledge about the system. This knowledge will grow by creating and collecting more and more classifiers. It is then possible to gain more information about the system by classifying the classifiers, i.e. by analyzing the basic properties of classifiers with, for example, the same clustering procedure that separated the population. This can be done in an iterative process (Fig. 1).

Mesoscopic Analysis

The mesoscopic analysis is a tool to find groups in data. The field of data analysis provides an overwhelming amount of methods, ranging from classical statistical methods (e.g., correlation analysis, c-means clustering) to modern adaptive methods from the fields of computational intelligence (e.g., neural networks, genetic programming). Furthermore, many more complex methods exist that are specialized for certain problem domains.

We do not present a new classification or clustering method here, but we suggest a method how to

[1]The classifiers introduced here are not to be confused with Holland's classifier systems.

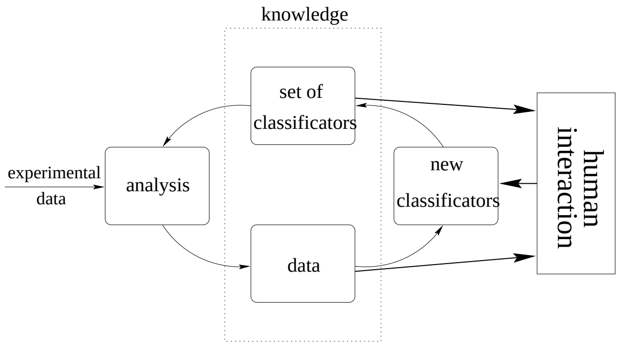

Figure 1: Iterative increase of knowledge.

apply available data analysis methods on artificial chemistries or on other population-based evolutionary artificial life systems. Our method will allow the integration of different available data analysis tools in an uncomplicated way.

First-Order Classifier: Property

A **first-order classifier** C relates an object s to a property p by indicating to what extend s has the property p.

A first-order classifier is defined as a map $C : S \longrightarrow [0, 1]$. Without loss of generality and for simplification the output of a classifier is normalized. If the output of $C(s) = p$ is either 0 (s does not have the property p) or 1 (p is property of s) we call the classifier **binary**. If p is continuous the classifier C might be called **fuzzy**.

There are different ways of constructing a classifier: (1) manually by estimating the classifying property by an apriori analysis of the data, and (2) automatically by learning of the mapping C. Typical examples are artificial neural networks, evolutionary algorithms or any other supervised or unsupervised algorithm with the ability to learn the classifying function C.

A typical example for a manually defined classifier is a detector C_s for a certain subsequence s. For the binary case $C_s(s')$ is equal to 1 if the subsequence s can be found in s' otherwise $C_s(s')$ is 0.

Second-Order Classifier: Distance

A **second-order classifier** is defined as a mapping $C : S \times S \longrightarrow [0, 1]$. It relates two objects. A classifier of this type is called a **distance measure** or shortly **distance**. If the conditions

$$\begin{aligned} C(s, s'') &\leq C(s, s') + C(s', s''), \\ C(s, s') &= C(s', s) \\ C(s, s) &= 0 \end{aligned} \tag{1}$$

are valid, the classifier C is a metric. As mentioned in the previous section we can define C manually. Hamming distance, for instance, is a well known distance measure. Using an evolutionary algorithm for the automatic generation of classifiers of any order is an appropriate method. Its fitness function then relies on the phenotypic/genotypic behavior of the objects or is given explicitly by user interaction (Banzhaf, 1997).

It is also possible to construct a second-order classifier C_F based on first-order classifiers $C_1, \ldots C_n$ by using the Euclidean distance

$$C_F(s, s') = \sqrt{\sum_i (C_i(s) - C_i(s'))^2} \tag{2}$$

or by methods taken from fuzzy set theory, i.g. (Miyamoto and Nakayama, 1986):

$$C_F(s, s') = \frac{\sum_{i=1}^{n} min(C_i(s), C_i(s'))}{\sum_{i=1}^{n} max(C_i(s).C_i(s'))} \tag{3}$$

Application and Integration of Classifiers

In this section we shall describe and demonstrate how classifiers may be integrated into a mesoscopic analysis of an evolutionary population-based system. The development of the population $P = (s_1, \ldots, s_M)$ in time requires the analysis of $P(t)$ (population at time t) for every single time step. The results of consecutive analyses have to be compared and correlated.

Clustering

The task of a cluster analysis procedure is to classificate a set $P = \{s_1, \ldots, s_n\}$ into c partial quantities (Bacher, 1996; Dunn, 1974). Each of these quantities is represented by a so-called **prototype** v_i, $(i = 1, \ldots, c)$. The degree of affiliation u_{ik} of object s_i to cluster c_k is computed and quantifies the probability of s_i to belong to c_k. In the case of fuzzy clustering u_{ik} is $\in [0, 1]$..

The partition of P should obey the following rules:

- similar data should be partitioned into similar clusters: **homogeneity**.
- different data should be separated into different clusters: **heterogeneity**.

The first step in a mesoscopic analysis could be a clustering based on a manually designed first-order classifier. The separating feature is, to what extend the objects have this property.

The second step could be a second-order classificator-based clustering. The number of clusters now depends on the internal structure of the population and the used classifier. The minimum of the objective function

$$J(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} \cdot d_{ik}^2 \tag{4}$$

with d_{ik}^2 symbolizing the distance between s_k and prototype v_i according to the used distance measure (a second-order classifier) is then the optimal clustering. Our distance measure for the following experiments is the Hamming distance, a natural choice given the binary nature of strings used as individuals here.

Distance

Classification of a population with a clustering algorithm thus depends crucially on the structure and dimension(s) of the sequence space.

The genotypic Hamming distance measures the number of different bits between two individuals represented by binary strings. Therefore the clustering only uses structural information to separate the population into different clusters. But the use of the genotypic information only causes in fact a loss of information.

Cluster algorithms using distance measures depending on both, genotypic and phenotypic information may be able to represent the internal structure of the population more accurately. The phenotypic distance between two individuals could be quantified by taking their functional behavior into account or by determining the difference between their fitness values according to a globally defined fitness function. If this global fitness function furthermore changes with an evolving population, it may be considered an **environmental distance**. This distance measure is a classifier of an order equal to the population size M.

Both, phenotypic and environmental distance are still under investigation. The following experimental results use genotypic distance only.

Tracing of Cluster Development

The development of a population is traced by analyzing its state after every generation (about 10^6 reactions). Each resulting cluster is represented by its specific prototype v_c which is the center of a hypersphere in distance space. The relationship between clusters at t_i and t_{i+1} is determined by the distances between their prototypes v_{c,t_i}, $v_{c,t_{i+1}}$.

An analysis of the development of properties of an artificial chemistry using the above explained techniques is the main object of the following sections.

Artificial Chemistry

Our mesoscopic analysis method will be applied to a static and dynamic analysis of an artificial chemistry (Varela, 1978; Lugowski, 1989; Rasmussen et al., 1990; Bagley et al., 1992; Fontana, 1991; Thürk, 1993; Bagley and Farmer, 1992; Banzhaf, 1993; Fontana and Buss, 1994). Usually, an artificial chemistry consists of at least two parts: 1.) a set of objects (molecules, substances) S and 2.) a set of collision rules. In addition, a simulation of the artificial chemistry requires a third component: an algorithm, which models the reaction vessel and is therefore called **reactor algorithm**.

In our case the objects are binary strings of fixed length $S = \{0,1\}^{32}$ (Banzhaf, 1993; Banzhaf, 1995). The collision rules are all second-order catalytic reactions of the form $s_1 + s_2 + X \longrightarrow s_1 + s_2 + s_3$, shortly $s_1 + s_2 \Longrightarrow s_3$. All collisions of two objects s_1, s_2 will have a unique outcome s_3. Therefore the set of collision rules can be represented as a function $r : S \times S \longrightarrow S$.

The following reactor algorithm operates on a population $P = \{s_1, \ldots, s_M\}$ which is a multiset on S.

Reactor algorithm

The development of the population P is realized by iteratively applying steps 2, 3, and 4 of the following reactor algorithm:

1. Initialize the population P with M objects selected randomly from S.
2. Select two objects s_1, s_2 from the population P randomly, without removing them.
3. If there exists a reaction $s_1 + s_2 \Longrightarrow s_3$ and the filter condition $f(s_1, s_2, s_3)$ holds, replace a randomly selected object of the population by s_3.
4. Goto step 2.

The filter condition $f : S \times S \times S \longrightarrow \{true, false\}$ is used to introduce elastic collisions easily, without changing the reaction mechanism. In our case f is defined as

$$f_1(s_1, s_2, s_3) = (s_1 \neq s_2 \wedge s_1 \neq s_3). \tag{5}$$

This filter condition inhibits the reaction if, on the one hand, operator s_1 and operand s_2 are the same and, on the other hand, if the operator string s_1 is an active replicator. (See Sec. *Automata Reaction* for more details.)

The reactor algorithm simulates mass-action kinetics of second-order catalytic reactions which allows hypercyclic dynamics (Eigen and Schuster, 1979; May, 1991). For a large population size M the system can be modeled by coupled ordinary differential

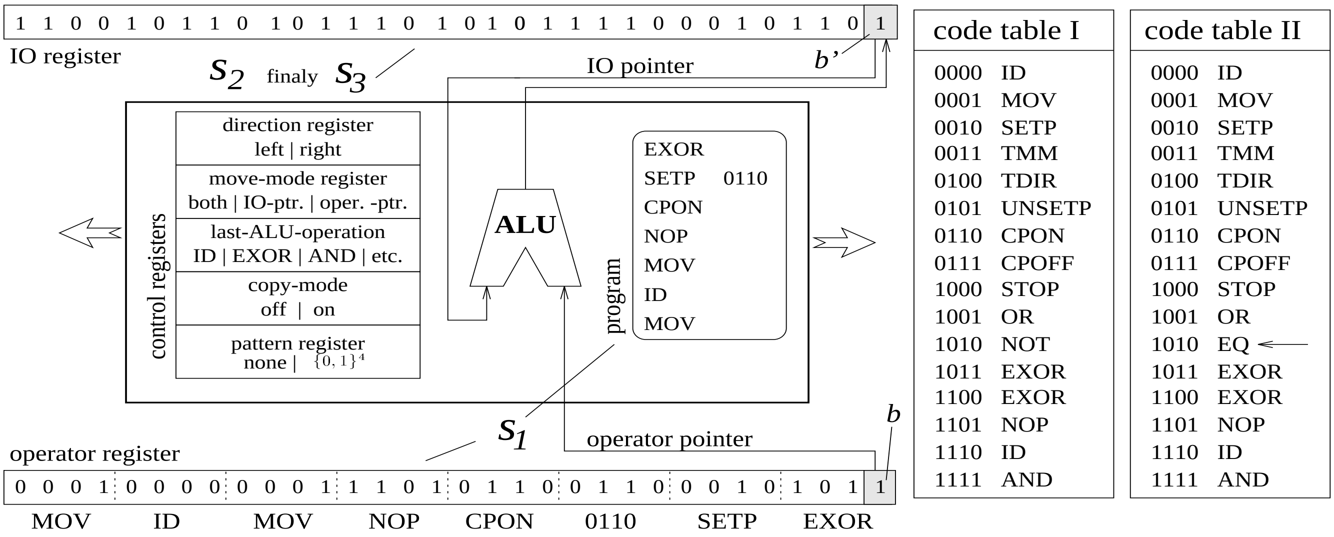

Figure 2: Automaton, resulting by folding s_1. It carries out the reaction $s_1 + s_2 \Longrightarrow s_3$. s_1 is written into the operator register and specifies the program. The IO register is initialized with s_2 and contains the result s_3 after running the program.

equations (Hofbauer and Sigmund, 1984; Stadler et al., 1993).

The following macroscopic measurements are used in the diagrams:

The **diversity** is the number of different strings in P divided by the population size M for normalization.

The **productivity** is the probability, that a collision of two strings is reactive, The term **collision** refers to one execution of steps (2) and (3) of the reactor algorithm. A collision is called **reactive** if a product s_3 is inserted into the population. So, a collision of two objects s_1, s_2 is reactive, if a product is defined by the reaction rule and if the filter condition allows the insertion of the product.

The **innovativity** is the probability of a collision producing an object that has never been in the system before.

We will use the the following 32-bit automata reaction as a reaction mechanism.

Automata Reaction

The **automata reaction** is based on a finite state automaton which is a mixture of a Turing-machine and a register machine. It has also been inspired by Hofstadter's (Hofstadter, 1985; Morris, 1989) Typogenetics.

The **automata reaction** instantiates a deterministic reaction $s_1 + s_2 \Longrightarrow s_3$, with $s_1, s_2, s_3 \in \{0,1\}^{32}$. In order to calculate the product s_3, string s_1 is "folded" into an automaton A_{s_1}, which gets s_2 as an input. The construction of A_{s_1} ensures that the automaton will halt after a finite number of steps. Because A_{s_1} is a deterministic finite automaton, the automata reaction defines a functions $\{0,1\}^{32} \times \{0,1\}^{32} \longrightarrow \{0,1\}^{32}$. Figure 2 shows the structure of the automaton. It contains two 32-bit registers, the **IO register** and the **operator register**. At the outset operator string s_1 is written into the operator register and operand s_2 into the IO register. The program is generated from s_1 by simply mapping successive 4-bit segments into instructions. The resulting program is executed sequentially, starting with the first instruction. There are no control statements for loops or jumps in the instruction set[2].

Each 32-bit register has a pointer, referring to a bit location, the **IO pointer**, referring to bit b' in the IO register and the **operator pointer**, referring to bit b in the operator register. Bit b and b' are inputs to the **ALU**. The result of the execution of one instruction of the operator register is stored at the IO pointers location, therefore replacing b'.

Instead of going into more details here we point the reader to a precise formal specification of the automata reaction as source code, available from (Dittrich, 1997) and a discussion of self-evolution in artificial chemistries based on the automata reaction in (Dittrich and Banzhaf, 1998).

As a short summary the following basic properties of the automata reaction should be noted:

[2]The instruction set used here is ID, MOV, SETP, TMM, TDIR, UNSETP, CPON, CPOFF, STOP, OR, EQ, EXOR, NOP, ID, AND. The resulting reaction is usually called a2-reaction

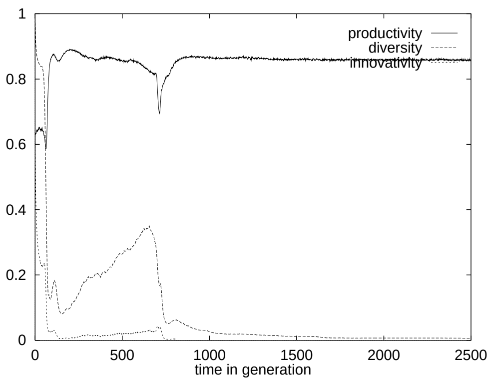

Figure 3: Diversity, productivity and innovativity of experiment A4-23 with the automata reaction a2. Parameters: $M = 100000$, reaction a2, no replication, full system seeding.

- The probability of producing a string s_3 similar to one of the colliding strings is high.
- The product of two randomly generated strings s_1, s_2 is likely ($p \approx 30\%$) equal to s_2. This is called **passive replication**, because the operator string s_1 does not modify the operand string s_2.
- A string s_1 for which $\forall x \in S : s_1 + x \Longrightarrow s_1$ is called **active replicator**, because it copies the operator string (itself) into the IO register. Active self-replicatiors are rare and have to be evolved during the development of the population. Due to their proliferation, they are the strings for which part two of the filter condition holds.

The structure of the population thus changes with the replicating ability of its individuals. Passive replicators may survive if the population is small ($M << 1000$) but they are displaced if active replicators evolve during a run. Due to the starting position of the register-pointers during a collision, the center positions of the string are likely to remain nearly unchanged while the possibility of changing the margins is higher.

Application

We will now demonstrate the application of the mesoscopic analysis method with data obtained from an artificial chemistry.

Figure 3 shows run A4-23 out of a series of 100 experiments all with the following parameter setting: population size: $M = 10^5$ (constant), reaction: automata reaction, reaction type: catalytic second-order, filter condition: active replication disabled (elastic), initialization: random strings (full system seeding). The observed behaviors in the series were very diverse ranging from early stabilization with short transients to complex, oscillating dynamics (Fig. 10). In (Dittrich and Banzhaf, 1998) we have identified two different evolutionary phases. The first one is characterized by high constructive activity which creates many new strings per generation. The second evolutionary phase begins with the emergence of a very stable, self-replicating core-set of cooperating strings dominating the system. This stable core-set keeps evolving by detachment of sub reaction pathways, by integration of totally new molecules or by emergence of new substrings which proliferate in almost every string. Here we will concentrate on one phenomenon in run A4-23: The sudden decrease and sudden increase of productivity around $t = 700$. The macroscopic observables in Fig. 3 are indicating that something is happening. The increasing diversity and the decreasing productivity indicate an interesting developmental phase. Even looking at the innovativity does not elucidate the process. So what happens and why? In principle we can reconstruct every single step and thus may be able to understand the phenomenon. But reducing the interval of interest to $t = 600$ until $t = 800$ still leaves about $16 * 10^6$ collisions between $3 \cdot 10^5$ different strings at $t = 600$.

We now start our mesoscopic analysis by asking the question:

(Q1)*Does the structure of the population change from $t = 600$ to $t = 800$?*

To answer this question we generated automatically (using genetic programming (GP) (Koza, 1992; Banzhaf et al., 1998)) a first-order classifier C_1 which is able to discriminate strings at $t = 600$ from strings at $t = 800$.

The training cases for the GP fitness function are the 400 most frequent strings of both, generation 600 and generation 800: $C_1(s)$ should be 0 if s is a string of gen. 600 and 1 if s is a member of the population at gen. 800.

The GP system has been used with the following settings: Operator set: conventional boolean functions, two ADFs (automatically defined functions) allowed. Random constants out of $\{0, 1, \ldots, 32\}$. $(\mu + \lambda)$ selection (Schwefel, 1995) with $\mu = 500$ (population size) and $\lambda = 350$ (number of descendents), finite life span: 5 generations. Program structure: tree and linear.

Among many other programms, GP created a short program which only tests bit 14 of the input string. Surprisingly, this bit discriminates not only the test cases but nearly every string in gen. 600 from any string in gen. 800. Thus, with the automatically generated classifier C_1 we are able to visualize the moment

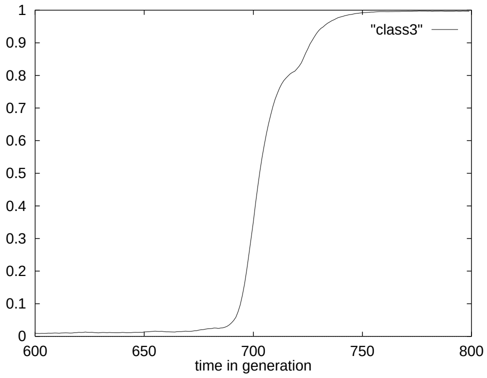

Figure 4: Normalized sum of classifier C_1 over time. Run A4-23.

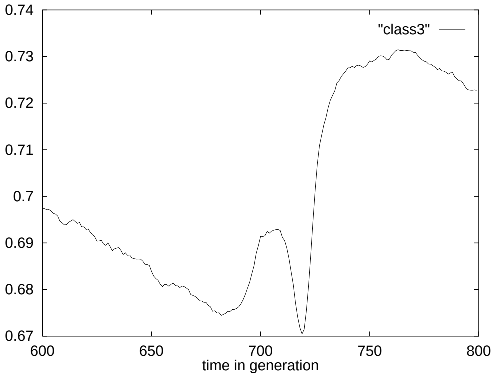

Figure 6: Normalized sum of classifier C_2 over time (self-replication ability). Run A4-23.

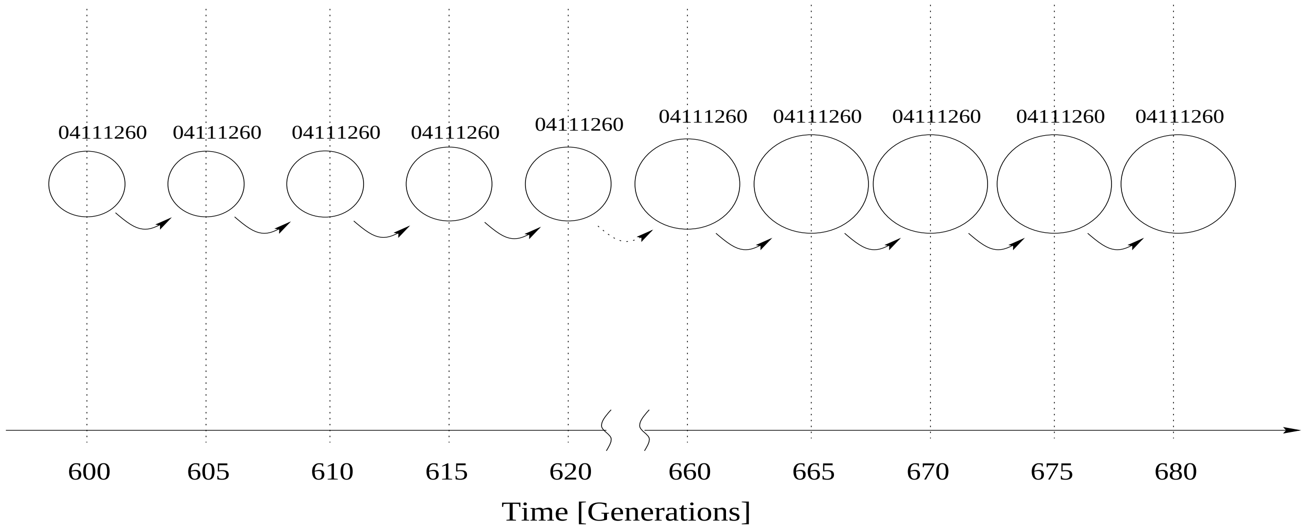

Figure 5: Clusterdevelopment of cluster **04111260**. Cluster method: fuzzy c-means. Run A4-23. 1 cm diameter $\approx$ 30 % of population.

when the population is beginning to change its structure (Fig. 4).

We have now identified a structural change, but

(Q2) *Does the functional property of strings change?*

To answer this question we define a first-order classifier C_2 manually:

$$C_2(s) = 1 - \frac{d_{Ham}((s \oplus s_0), s) + d_{Ham}((s \oplus s_f), s)}{64} \tag{6}$$

where s_0 =**0000000** and s_f =**fffffff**. This classifier estimates the self-replicating ability by testing s against the s_0 and the s_f string. The string s is the operator processing both, s_0 and s_f. The number of bit flips, either from 0 to 1 ($d_{Ham}((s \oplus s_0), s)$) or from 1 to 0 ($d_{Ham}((s \oplus s_f), s)$) caused in s_0 and s_f indicate the replicating ability of s. $C_2(s)$ is at maxixmum if s causes as many bit flips in s_0 (and s_f) as necessary to reproduce its bit pattern. $C_2(s)$ is 0, if s does not change neither s_0 nor s_f. Indeed, Fig. 6 shows that there is a functional change. The replicating ability of strings in the population decreases until $t \approx 690$. There is a strong increase in C_2 from $t = 720$ until $t = 750$, the point in time at which the restructuring of P seems to be finished. The questions to answer are now:

(Q3) *How is the population structured at* $t = 600$*?*
(Q4) *What happens at* $t = 690$*?*

The structure of the population in experiment A4-23 at about $t = 600$ until $t = 680$ is shown in Fig. 5. All strings present belong to cluster **04111260**, so their

Figure 7: Clusterdevelopment. Clustering performed on the 400 most frequent strings per generation. 1 cm diameter $\approx$ 20% of population. Cluster method: fuzzy c-means. Run A4-23.

genotypic structure is almost the same. There is a slight increase in volume which indicates that more and more individuals are replaced by one of the most frequent 400 strings. Figure 7 shows a second-order cluster analysis of these 400 most frequent strings from $t = 693$ to $t = 699$, the moment of the supposed restructuring. At $t = 695$ the single cluster **0c211260** (**04111260** of Fig. 5 slightly changed its center) splits into two clusters **4b0112d6** and **4b1412d6**. The former disappears after two generations while the latter quickly grows and changes its center (in $t = 699$ almost half of the population is in **4b1432d4**). Analysis shows that the center of the shrinking cluster has almost the same structure as in previous generations but the second cluster has an obviously different bit pattern on the left side. Further development of the remaining cluster shows that now both, individuals with modified and unmodified left and right sides belong to the same cluster. The point of interest is now:

(Q5) *Is there a change in the way genotypic information is reproduced or conserved ?*

Looking at the clusters obtained from (Q3) and (Q4) reveals that strings in the displaced cluster have high diversity on the left side. Strings in the new cluster are much more homogeneous on the left side. This raises the question whether the members of the new cluster have acquired the ability to replicate their left side more accurately. To validate this assumption we have

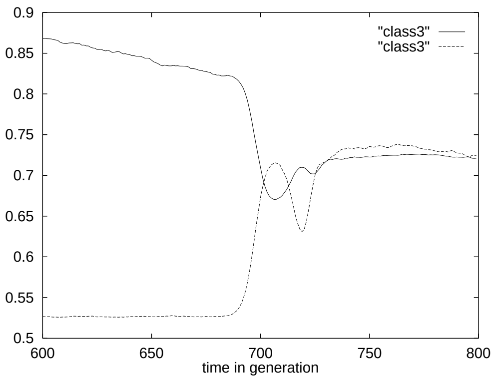

Figure 8: Normalized sum of classifier $C_{0000ffff}$ and $C_{ffff0000}$ over time. Run A4-23.

defined a first-order classifier C_{s_m} which checks for replicating activity at certain bit positions specified by the substring s_m:

$$\begin{aligned} C_{s_m}(s) = 1 \quad &- \quad \frac{d_{Ham}(((s \oplus s_0) \wedge s_m), s \wedge s_m)}{2\#ones(s)} \\ &- \quad \frac{d_{Ham}((s \oplus s_f) \vee \neg s_m, s \vee \neg s_m)}{2\#ones(s)} \end{aligned} \quad (7)$$

with $s_m \in S$ and $\#ones(s)$ denotes the number of ones in s. (Note that $C_2 = C_{s_m}$ with $s_m =$ **ffffffff**.)

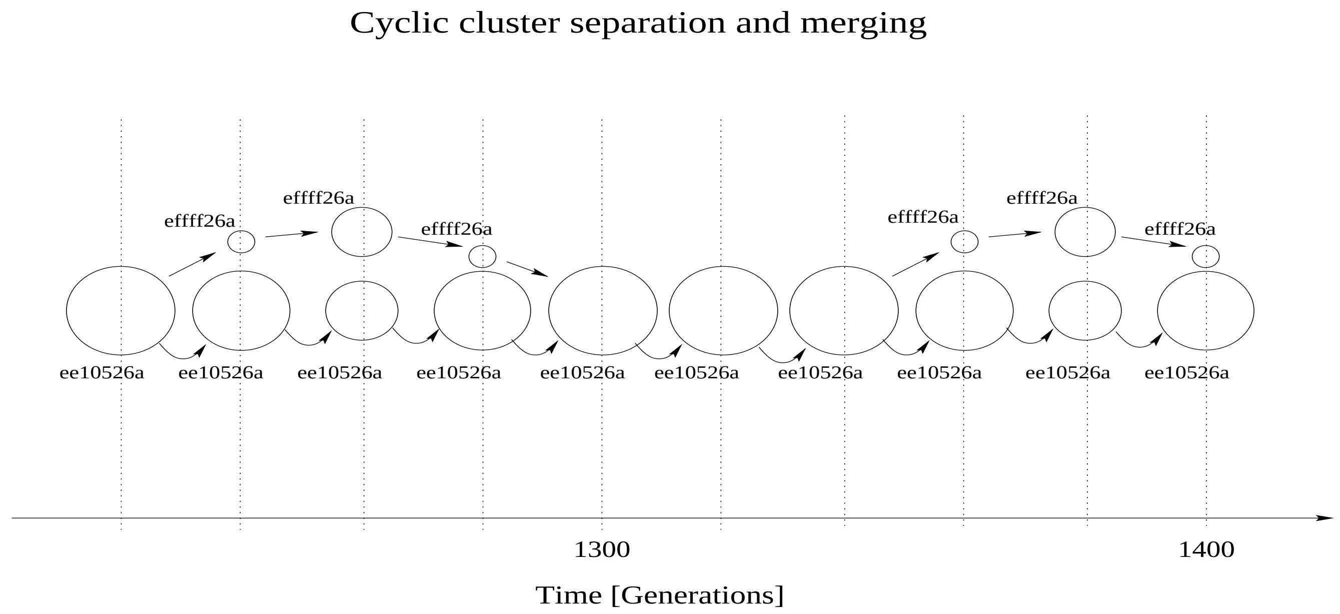

Figure 9: Time development of clusters **ee10526a** and **effff26a** in run A4-49. 1 cm diameter $\approx$ 20% of population. Cluster method: fuzzy c-means.

The change of replication from right-oriented to left and-right-oriented is shown in Fig. 8 by showing the development of $C_{0000ffff}$ and $C_{ffff0000}$.

Another very interesting experiment is shown in Fig. 10. Obviously, there is an oscillating behavior of the the macroscopic parameters from $t = 1000$ to $t = 2000$. A zoom into the interval from $t = 1200$ to $t = 1600$ is shown in Fig. 11. This run is remarkable because its complex behavior is an exception. Most of the experiments are similar to run A4-23, about 20% are even simpler. The result of a cluster analysis of A4-49 is shown in Fig. 9. Here, an oscillating separation and merging of two clusters with stable centers (**ee10526a** and **effff26a**) is the dominating phenomenon.

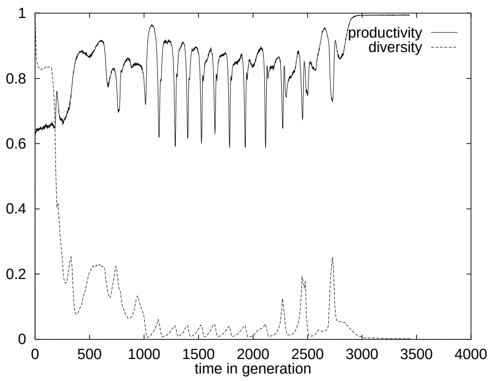

Figure 10: Diversity, productivity and innovativity of experiment A4-49 with the automata reaction. Parameters: $M = 100000$, no replication.

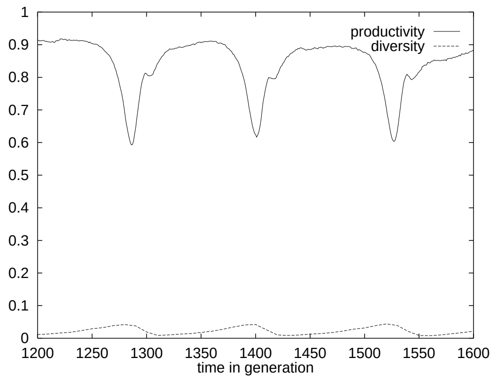

Figure 11: Magnification of Fig. 10.

Discussion

A new mesoscopic step-wise analysis method for evolutive population-based systems has been suggested. Knowledge about the system is gained by decomposing it with the help of classifiers. The application of classifiers generates information which can be used to generate new classifiers which in turn can be applied to the system. This concept allows the integration of different automatic and manual analysis techniques. The method has been successfully applied to an artificial chemistry. A sudden decrease and increase of a macroscopic quantity (productivity) has been identified as the indication of the appearance of a new organization (cluster). Its development has been visualized and investigated by dynamic cluster analysis. The resulting clusters motivated the definition of more specialized classifiers which confirmed the assumption that the new organization has acquired the ability to replicate the left side of a string.

Future Work

Classifiers can be generated automatically (i.e. C_1), may be parameterized (i.e. C_{s_m}) or may be combined into a huge number of new classifiers. To ease the handling of classifiers we can assign a "meaning" to each classifier, e.g., a text describing the properties of the classifier, and set up a data base containing the classifier name, its formal definition (program) and its meaning (text). We are then able to combine and integrate formal descriptions with more intuitive, linguistic descriptions. In a second step the classifiers may be correlated.

If we also allow classifiers to be discarded (for instance, because they are redundant measures), the resulting process seems to become an evolutionary process, now on the level of classifiers. So, we can apply the same techniques as described here not only to the primary population system but also to the system of evolving classifiers.

Acknowledgements

This project is supported by the DFG (Deutsche Forschungsgemeinschaft), grant Ba 1042/2-2. We would like to thank Christian Düntgen, Ahmet Koç, Andre Skusa, Markus Brameier and Wolfgang Kantschik.

References

Bacher, J. (1996). *Clusteranalyse.* Oldenbourg, München, Wien.

Bagley, R. J. and Farmer, J. D. (1992). Spontaneous emergence of a metabolism. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Proceedings of the Workshop on Artificial Life (ALIFE '90)*, volume 5 of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 93–140, Redwood City, CA, USA. Addison-Wesley.

Bagley, R. J., Farmer, J. D., and Fontana, W. (1992). Evolution of a metabolism. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Proceedings of the Workshop on Artificial Life (ALIFE '90)*, volume 5 of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 141–158, Redwood City, CA, USA. Addison-Wesley.

Banzhaf, W. (1993). Self-replicating sequences of binary numbers – foundations i and ii: General and strings of length n = 4. *Biological Cybernetics*, 69:269–281.

Banzhaf, W. (1995). Self-organizing algorithms derived from rna interaction. In Banzhaf, W. and Eeckman, F. H., editors, *Evolution and Biocomputation: Computational Models of Evolution*, pages 69–102. Springer, Berlin-Heidelberg-New York.

Banzhaf, W. (1997). Interactive evolution. In Bäck, T., Fogel, D. B., and Michalewicz, Z., editors, *Handbook of Evolutionary Computation.* IOP Publishing.

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. (1998). *Genetic Programming - An Introduction.* Morgan Kaufmann and dpunkt Verlag.

Dittrich, P. (1997). Source code of the automata reaction. Internet FTP server at the Chair of Systems Analysis, Dept. of Comp. Science, Univ. of Dortmund, file name: *ftp://lumpi.informatik.uni-dortmund.de/pub/biocomp/src/autoreac-1.0.tar.gz.*

Dittrich, P. and Banzhaf, W. (1998). Self-evolution in a constructive binary string system. Artificial Life, in press.

Dunn, J. C. (1974). Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4:95 – 104.

Eigen, M. and Schuster, P. (1979). *The Hypercycle.* Springer, Berlin.

Fontana, W. (1991). Algorithmic chemistry. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II: Proceedings of the Second Artificial Life Workshop*, pages 159–209. Addison-Wesley, Reading MA.

Fontana, W. and Buss, L. W. (1994). "The arrival of the fittest": Toward a theory of biological organisation. *Bull. Math. Biol.*, 56:1–64.

Hofbauer, J. and Sigmund, K. (1984). *Evolutionstheorie und dynamische Systeme.* Paul Parey, Berlin.

Hofstadter, D. R. (1985). *Gödel, Escher, Bach:.* Klett-Cotta, Originalausgabe: 1979, Basic Books, New York, 6. edition.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Natural Selection.* MIT Press, Cambridge, MA, USA.

Lugowski, M. W. (1989). Computational metabolism: Towards biological geometries for computing. In

Langton, C. G., editor, *Artificial Life*, pages 341–368. Addison-Wesley.

May, R. M. (1991). Hypercycles spring to life. *Nature*, 353:607–608.

Miyamoto, S. and Nakayama, K. (1986). Similarity measures based on a fuzzy set model and application to hieracical clustering. *IEEE Trans., Syst., Man and Cybernetics*, 16(3):479–482.

Morris, H. C. (1989). Typogenetics: A logic for artificial life. In Langton, C. G., editor, *Artificial Life*, pages 369–395. Addison-Wesley.

Rasmussen, S., Knudsen, C., Feldberg, R., and Hindsholm, M. (1990). The coreworld: Emergence and evolution of cooperative structures in a computational chemistry. *Physica D*, 42:111–194.

Ray, T. S. (1992). An approach to the synthesis of life. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Proceedings of the Workshop on Artificial Life (ALIFE '90)*, volume 5 of *Santa Fe Institute Studies in the Sciences of Complexity*, Redwood City, CA, USA. Addison-Wesley.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology Series. John Wiley & Sons, Inc., New York.

Stadler, P. F., Fontana, W., and Miller, J. H. (1993). Random catalytic reaction networks. *Physica D*, 63:378–392.

Thürk, M. (1993). *Ein Modell zur Selbstorganisation von Automatenalgorithmen zum Studium molekularer Evolution*. PhD thesis, Universität Jena, Naturwissenschaftliche Fakultät.

Varela, F. (1978). On being autonomous: The lessons of natural history for systems theory. In Klir, G., editor, *Applied General Systems Research*, pages 77–84.