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 structures and the interactions (reactions) among them are defined by an algorithm. The same object can appear in two forms: (1) as an active machine (operator), (2) as passive data (operand). Thus, the same object can act on other objects or it can be processed by others. This dualism allows the implicit definition of constructive artificial chemical systems, which exhibit quite complex behaviors. In our case even evolutionary behavior has been observed which is notable, because no explicit mutation, recombination or fitness function is involved. Every variation is exclusively performed by the objects (molecules) in their machine form. In addition to microscopic methods (e.g. monitoring the actions of single molecules) and macroscopic measurements (e.g. diversity, complexity) we developed a stepwise mesoscopic analysis method based on classification and dynamic clustering. Knowledge about the system is accumulated in a cyclic process, where measuring tools (classifiers) extract information, which is again used to create new classifiers.

Keywords:

self-organization, strong artificial life, evolution, computational chemistry, visualization, 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. Analysis is often much harder, especially in population based systems where a huge number of diverse individuals are interacting. In this case the reduction of experimental data is required. Measuring only macroscopic parameters (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 (i.e. $O(n)$) is lower than time complexity needed for analysis (i.e. $O(n \log(n)$ for sorting the population). Therefore, methods for analysis in between macroscopic and microscopic methods are required.

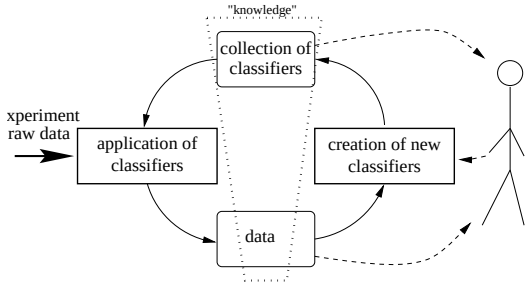

Figure 1: Cyclic increase of knowledge.

In this paper we present a stepwise mesoscopic analysis method for artificial chemistries, which might as well be applied to other population-based systems. In the mesoscopic analysis the population is decomposed into groups. The decomposition is based on so called **classifiers**. A classifier represents a property of an object. A first-order classifier is simply a function $C : S \longrightarrow [0, 1]$. The value $C(s)$ 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 cyclic process (Fig. 1).

The mesoscopic analysis introduced here will be demonstrated in a specific artificial chemistry.

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 (i.e. correlation analysis, c-means clustering) to modern adaptive methods from the fields of computational intelligence (i.e. neural networks, genetic programming). Furthermore, many more complex methods exist specialized for certain problem domains.

Here, we do not present a new classification or clustering method, but we suggest a method how to apply available data analysis methods on artificial chemistries or on another population-based evolutionary artificial life system. 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 to a property. From an object oriented point of view we can say that a classifier represents a property and is able to indicate to what extend an object has this property. 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 is either 0 or 1 we call the classifier **binary**. If it is continuous it might be called **fuzzy**.

There are different ways of constructing a classifier: (1) manually by estimating the classifying property through an apriori analysis of the data, and (2) automatically by learning of a mapping C. Typical examples are artificial neural networks or evolutionary algorithms or any supervised or unsupervised algorithm with the ability to create 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 s can be found in s' otherwise 0.

Second-Order Classifier - Distance

A **second-order classifier** is defined as a mapping $C : S \times S \longrightarrow [0, 1]$. It relates one object to another. A second-order classifier can also be called a **distance measure** or shortly **distance**. If the condition $C(s, s'') \leq C(s, s') + C(s', s'')$ is valid and $C(s, s) = 0$, the classifier C is a metric.

As in the previous section we can define C manually. A well known distance measure is for example the Hamming distance.

For an automatic generation the application of an evolutionary algorithm would be plausible. Its fitness function could rely on the phenotypic/genotypic behavior of the objects or could be given implicitly 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 Euclidean distance

$$C_F(s, s') = \sqrt{\sum_i (C_i(s) - C_i(s'))^2} \qquad (1)$$

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

$$C_F(s, s') = \frac{\sum_{i=1}^{n} min(C_i(s), C_i(s'))}{\sum_{i=1}^{n} max(C_i(s).C_i(s'))} \qquad (2)$$

Application and Integration of Classifiers

In this section we will describe and demonstrate methods how classifiers can be integrated into a mesoscopic analysis of an evolutionary population system. The development of the population $P = (s_1, \ldots, s_M)$ in time needs the analysis of $P(t)$ (population at time t) for every single time step. The results of consecutive analysis have to be compared and put into relation.

Clustering

The task of a cluster analysis procedure is the classification of 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** $v_i, \quad (i = 1, \ldots, c)$. The degree of affiliation u_{ik} of object s_i to cluster c_k is computed and describes the probability of s_i to belong to c_k. In the case of fuzzy clustering u_{ik} is $\in [0, 1]$..

The division of P should obey the following rules:

- similar data should be distributed 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 typical number of clusters in this case is two. The separating feature is to what extend the objects have this property.

The second step could be a second-order classifier-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

$$J(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik}) * d_{ik}^2 \qquad (3)$$

with d_{ik}^2 the distance between s_k and the prototype v_i according to the used distance measure is then the optimal clustering. The distance measure for the following experiments is the Hamming distance, which is evident because of the binary strings representing the individuals in the artificial chemistry.

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. 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 better represent the internal structure of the population. The phenotypic distance between two individuals could be quantified by taking their functional behavior into account or by determing the difference between their fitness values according to a globally defined fitness function. If this global fitness function changes with an evolving population one could speak of an **environmental distance**.

Both phenotypic and environmental distance are still under investigation. The presented experimental results used the genotypic distance.

Tracing of Cluster Development

The development of the population could be traced by analyzing the state after every generation (about 10^6 reactions). The resulting clusters are represented by their specific prototype which is analogous to the center of a cluster in Euclidean space. The relationship between the segmentation at t_i and t_{i+1} is determined by the distances between their prototypes.

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; Bagley and Farmer, 1992; Fontana, 1991; Fontana and Buss, 1994; Th"urk, 1993). Usually, an artificial chemistry consists of at least two parts: 1.) a set of objects (molecules, substances) S, 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 $r : S \times S \longrightarrow S$, here.

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

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

$$f_1(s_1, s_2, s_3) = (s_1 \neq s_2 \wedge s_1 \neq s_3) \qquad (4)$$

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 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 if defined by the reaction rule and if the filter condition allows the insertion of the product.

The **innovativity** is the probability that a collision produces an object that is totally new in the system.

As a reaction mechanism we will use the the following 32-bit automata reaction.

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 $s_1 + s_2 \Longrightarrow s_3$, where $s_1, s_2, s_3 \in \{0,1\}^{32}$. 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 bounded 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 beginning 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 [1].

Each 32-bit register has a pointer, referring to a bit location. The **IO pointer**, referring to a bit b' in the IO register and the **operator pointer**, referring to a bit b in the operator register. Bit b and b' are inputs to the **ALU**. The ALU result 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

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

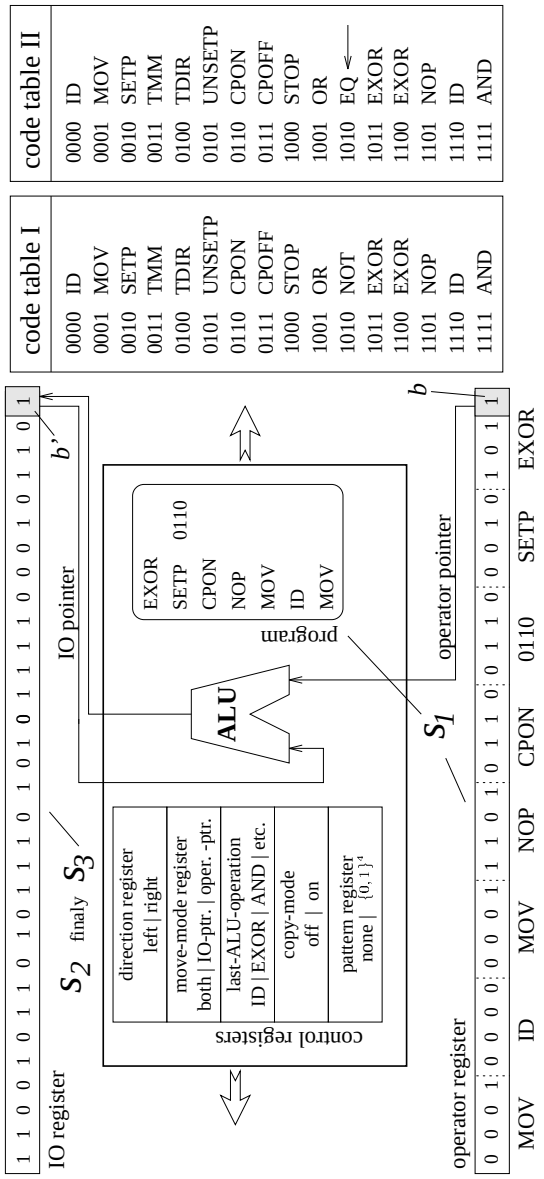

Figure 2: Automaton, resulting by folding s_1. It carries out the reaction $s_1 + s_2 \Longrightarrow s_3$. s_1 is written into the operator register and specifies the program. The IO register is initialized with s_2 and contains the result s_3 after running the program.

(Dittrich and Banzhaf, 1998).

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

- The probability is high that a product of a non-elastic collision is similar to one of the colliding strings.
- 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-replication is rare and has to be evolved during the development of the population.

The structure of the population thus changes with the replicating ability of it's 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 pointers during a collision, the center positions of the string are likely to remain nearly unchanged while the possibility of changing margins is higher.

Application

We will now demonstrate the application of the mesoscopic analysis method on 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 a2, reaction type: catalytic second-order, filter condition: exact 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. 8). 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

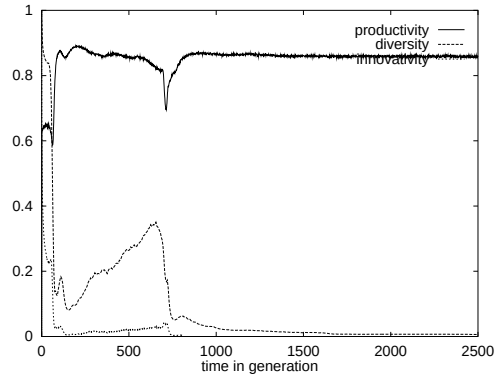

Figure 3: Diversity, productivity and innovativity of experiment A4-23 with the automata reaction a2. Parameters: $M = 100000$, reaction a2, no replication, full system seeding.

productivity around $t = 700$. The macroscopic observables in Figure 3 are indicating that something is happening. But what and why ? In principle we can reconstruct every single step and thus are able to understand the phenomenon. But although the interval is reduced for the analysis to $t = 600$ - $t = 800$ there are still about $16 * 10^6$ reactive collision events left which in addition, are very diverse because of the high number of different string types (31955 different strings at $t = 600$).

We start our 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 of $t = 600$ from strings of $t = 800$. The training cases for the GP fitness function where the 400 most frequent strings of each generations. $C_1(s)$ should be 0 if s is a string of gen. 600 and 1 if s is a member of the population at time 800. The GP system has been used with the following setting: 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 others 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 those in ge. 800. Thus, with the automatically generated classifier C_1 we are able to visualize the moment when the population is beginning to change its structure (Fig. 4).

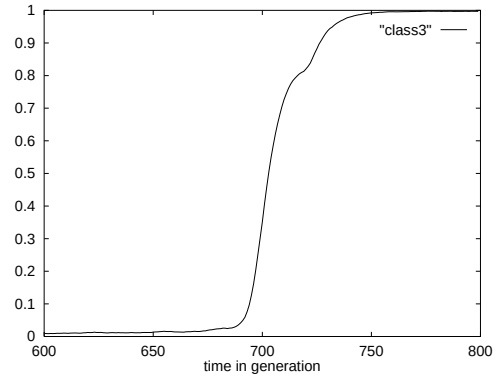

Figure 4: Normalized sum of classifier C_1 over time. Run A4-23.

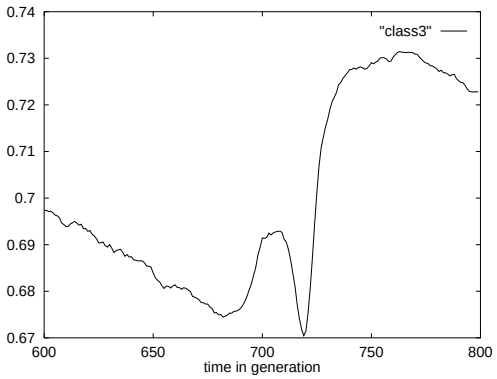

Figure 5: Normalized sum of classifier C_2 over time (self-replication ability). Run A4-23.

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 "by hand":

$$C_2(s) = 1 - \frac{d_{Ham}((s \oplus s_0), s) + d_{Ham}((s \oplus s_f), s)}{64} \tag{5}$$

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. Indeed, figure 5 shows with the help of C_2 that there is a functional change. Now questions to answer are

(Q3) *How is the population structured at* $t = 600$*?*

(Q4) *How does the cluster evolve?*

The structure of the population in experiment A4-23 at about $t = 600$ until $t = 680$ is shown in fig. 6. All present strings belong to cluster **04111260**, so their 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.

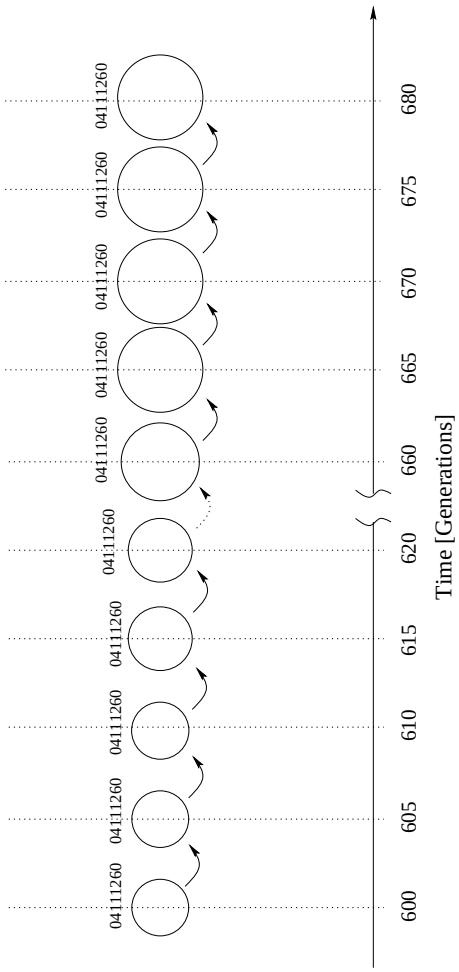

Figure 6: Clusterdevelopment of cluster **04111260**. Cluster method: fuzzy c-means. Run A4-23.

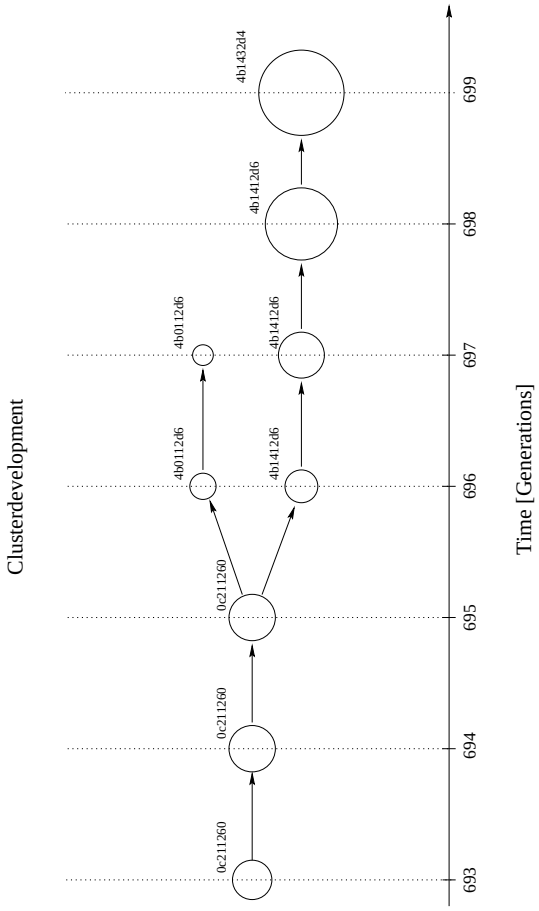

Figure 7: Clusterdevelopment. Clustering performed on the 400 most frequent strings per generation. 1 cm diameter $\approx$ 15% of population. Cluster method: fuzzy c-means. Run A4-23.

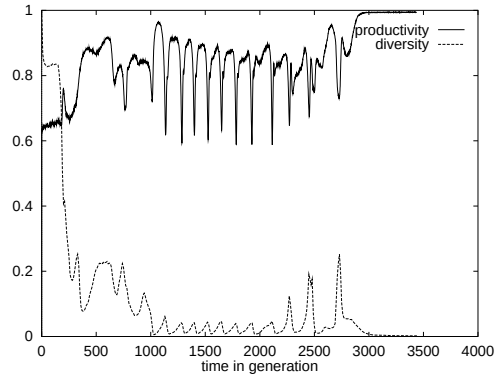

Figure 8: Diversity, productivity and innovativity of experiment A4-49 with the automata reaction. Parameters: $M = 100000$, reaction a2, no replication, full system seeding.

Figure 7 shows a second-order cluster analysis of these 400 most frequent strings from $t = 693$ to $t = 700$, the moment of the supposed restructuring. At $t = 695$ the single cluster **0c211260** (**04111260** slightly changed it's 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 before but that 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.

A very interesting experiment was run A4-49. As shown in fig. 8, there is an oscillating behavior of the the makroscopic parameters from $t = 1000$ to $t = 2000$. A magnification of the interval from $t = 1200$ to $t = 1600$ is shown in fig. 9. This run is remarkable because this quite complex behavior is an exception. Most of the experiments are similar to run A4-23n about 20% are even simpler. The result of a following cluster analysis is shown in fig. 10. Here an oscillating separation and merging of two clusters with stable centers (**ee10526a** and **effff26a**) is the dominating phenomenon.

(Q5) *Is there a change in the way genotypic information is reproduced/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 new cluster has acquired the

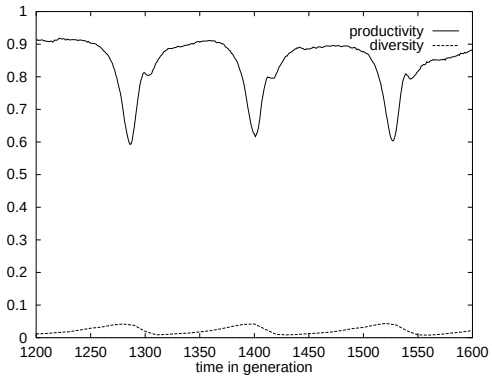

Figure 9: Magnification of the analysis interval for run A4-49.

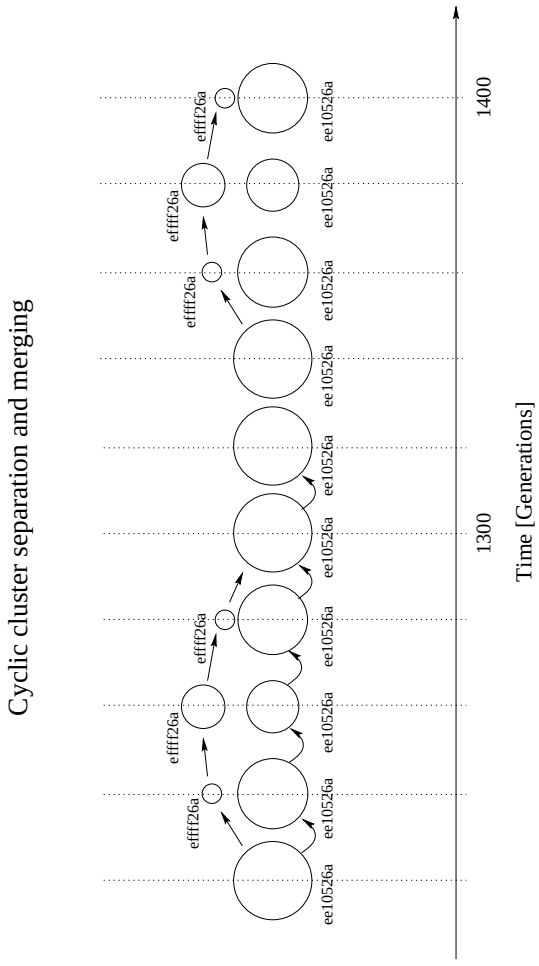

Figure 10: Time development of clusters **ee10526a** and **effff26a** in run A4-49. 1 cm diameter $\approx$ 20% of population. Cluster method: fuzzy c-means.

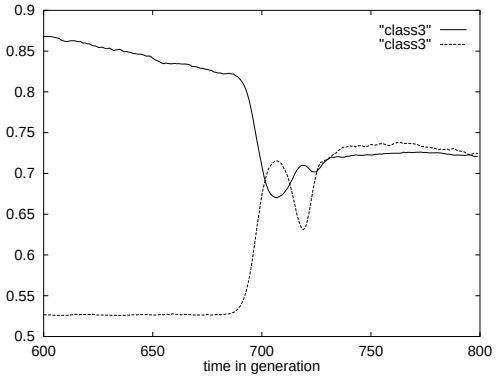

Figure 11: Normalized sum of classifier $C_{0000ffff}$ and $C_{ffff0000}$ over time, which measures the left-oriented replication and right-oriented replication ability, respectively. Run A4-23.

ability to replicate it's left side more accurately.

To validate this assumption we have defined a first-order classifier C_{s_m} which checks for replicating activity at certain bit positions specified by the mask s_m:

$$C_{s_m}(s) = 1 \quad - \quad \frac{d_{Ham}(((s \oplus s_0) \wedge s_m), s \wedge s_m)}{2\#ones(s)} \quad - \quad \frac{d_{Ham}((s \oplus s_f) \vee \neg s_m, s \vee \neg s_m)}{2\#ones(s)} \quad (6)$$

where $s_m \in S$ is a mask and $\#ones(s)$ denotes the number of ones in s. Note, that $C_2 = C_{fffffffff}$. The change of replication from right-oriented to left-and-right-oriented is shown in figure 11 by showing the development of $C_{0000ffff}$ and $C_{ffff0000}$.

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 can again be applied to the system. The concept allows the integration of different automatic and manual analysis techniques.

The method has been successfully applied to an articial chemistry. A sudden decrease and increase of a macroscopic quantity (productivity) has been identified as the birth of a new organization (cluster). Its development has been visualized and investigated by a dynamic cluster analysis. The resulting clusters motivated the definition of more specialized classifiers which confirm the assumption that the new organi-

zation has acquired the ability to replicate the left side of a string.

Future work: Classifiers can be generated automatically (i.e. C_1), can be parameterized (i.e. C_{s_m}) or can be combined which can lead quickly to a huge number of classifiers. To ease the handling of the classifiers we can assign to each classifier a "meaning" which is 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). With this we are able to combine and integrate formal descriptions with more intuitive, linguistic descriptions. In a second step the classifiers can be related to each other.

If we also allow classifiers to be discarded (i.e. because they are redundant), 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 on the primary population system, but also on the system of evolving classifiers.

Acknowledgements

This project is supported by the DFG (Deutsche Forschungsgemeinschaft), grant Ba 1042/2-2. We also thank Christian Düntgen, Ahmet Koç, Andre Skusa, Marcus 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 B"ack, 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 dounkt 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: `autoreac-1.0.tar.gz`.

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

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). *G"odel, 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.

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.

Th"urk, M. (1993). *Ein Modell zur Selbstorganisation von Automatenalgorithmen zum Studium molekularer Evolution*. PhD thesis, Universit"at Jena, Naturwissenschaftliche Fakult"at.

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.