How to Program Artificial Chemistries

Jens Busch and Wolfgang Banzhaf

University of Dortmund,
Department of Computer Science, Chair of Systems Analysis (LS XI)
D-44221 Dortmund, Germany
{busch, banzhaf}@ls11.cs.uni-dortmund.de
http://ls11-www.cs.uni-dortmund.de

Abstract. Using the framework of artificial chemistries (ACs) an automated theorem prover (ATP) is constructed. Though it is an application of its own, in the context of ACs automated theorem proving can serve a second purpose. In this paper, we present a resolution-based AC named *RESAC*. Once converted to the first-order predicate calculus a problem straightly fits to this non-deterministic AC model. The calculus therefore provides a general and intuitive language for "programming" *RESAC*. The fixed implicit interaction scheme and predefined structure of the objects is advantageous and helps to predict the system's dynamics. Furthermore, the versatility of the methodology is demonstrated by implementing the Adleman problem. An analysis of the dynamic behavior is performed delivering insight into the synthesis of non-deterministic emerging processes. This analysis include a discussion of some general AC parameters.

1 Introduction

Inspired by real chemical processes, artificial chemistries (ACs) follow an intriguing computational paradigm. Abstract molecules (objects) interact in an autonomous, distributed and parallel way. Guided by reaction rules these objects act on one another, thereby proliferating or altering their information content or structure. Modified structure may imply a different function according to the imposed interaction scheme. Consequently, the role of a molecule can change over time and this way, AC provides a powerful framework to define constructive systems [10]. The actual state of the system is defined by concentration levels of molecules. A calculation step itself causes a change of such concentrations.

Formally, an AC is defined by the triple (S, R, A) [11]. S denotes a set of objects and R a set of interaction rules constituting the reaction schemes applied whenever molecules collide. The system's dynamics is controlled by an algorithm A describing how the rules are applied to a population of molecules. By this means a closed or an open reactor can be simulated, a well-stirred reaction vessel with no topology or for example a 3-D space like in [19].

On the one hand, ACs try to answer questions arising in evolution theory. Here, the emergence of global phenomena in restricted environments initiated

W. Banzhaf et al. (Eds.): ECAL 2003, LNAI 2801, pp. 20–30, 2003.

by a huge number of reactions among elementary entities is investigated. Results have been presented e.g. in [3,4,6,12,13,16]. On the other hand, researchers link theory with practical applications by taking advantage of the similarity between ACs and other complex dynamic systems (e.g. [2,7,8,15,20]).

A natural definition for many tasks is in terms of elementary interactions to be performed frequently. But what "language" should be used to encode a given task? Even more troublesome is the choice of a molecule representation and of an interaction scheme if one wants to deal with many problems from various domains instead of one specific task. In this paper, we present first-order predicate calculus (FOPC) as a general and intuitive molecule representation. Combined with resolution logics as an implicitly defined interaction rule we gain the dynamic system *RESAC* which can be interpreted in two different ways: (i) As implementation of an ATP (AC-based ATP). FOPC problem solving belongs to the class of hardest known problems in computer science, since the validity of FOPC-formulas is not decidable. In this area previous work was done and results have been presented in [9]. (ii) As object and reaction rule setup for the solution of arbitrary tasks (ATP-based AC) by defining both sets S and R.

Here, we are concerned with the second alternative. Regarded as a tool, the framework of ATP provides RESAC with a fixed setting (S, R), because data structure and interaction mechanism are defined independent from a specific task. This simplifies the AC design and the analysis of the system's dynamics considerably. However, the specification of an algorithm A is still necessary. In Sect. 3 we suggest such an algorithm, discuss several variants, and construct RESAC. As an example, the famous Adleman problem is run in RESAC in Sect. 4, and its dynamics is analyzed.

2 Basic Concepts of Logic and ATP

ATP might be considered one of the oldest branches of Artificial Intelligence. Given a *theory* $A = \{A_1, \ldots, A_n\}$ of expressions the task is to proof that another expression T is consistent with theory A ($A \models T$). Of course, checking all possible truth-assignments (*interpretations*) will solve the problem. Unfortunately systems of predicates have a potentially infinite number of interpretations, therefore this is not a practical approach. The combinatorial explosion of search space demands another, more sophisticated search strategy. Having an eye on the syntax instead on the semantics can be helpful to restrict the search space: A syntactical transformation mapping one expression onto a new expression consistent with the underlying theory is called *sound inference rule*. If expression T is producible by a sequence of inference rules operating on both, the theory A and every expression inferred from A so far, then $A \models T$ holds and T is called *theorem* of A. The protocol belonging to the inference sequence is said to be the proof of $A \models T$.

To construct RESAC, we will use concepts of FOPC which provides a representation for objects, their properties and relations. Note that this calculus is much more powerful than calculi restricted to Horn clauses used in PROLOG-

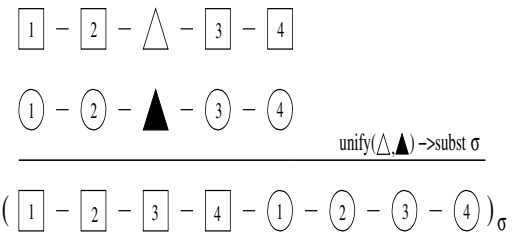


Fig. 1. Illustration of the resolution principle

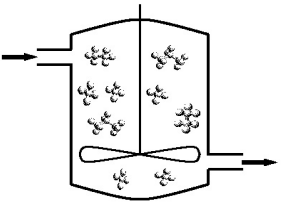


Fig. 2. Open, well-stirred reactor

based systems (e.g. [18]). In FOPC, logic expressions can be converted into an inferentially equivalent conjunction of *clauses* each being a disjunction of *literals*. These are either atomic expressions or negated atomic expressions. $|l|$ denotes the literal l without polarity.

Regarding RESAC, one inference rule is of special interest: binary resolution [17]. Given two clauses c_1 and c_2 with literals $l_1 \in c_1$ and $l_2 \in c_2$, a new clause is inferred by resolution if $|l_1|$,$|l_2|$ are unifiable[1] and of complementary polarity. The resolvent is obtained by joining c_1 and c_2 with the appropriate substitution applied and eliminating l_1 and l_2 (Fig. 1 outlines this scheme).

To be more effective, the *set-of-support strategy* may be applied: In this case, at least one of $\{c_1, c_2\}$ must have *support* in order to allow a resolution inference. Every result of an inference inherits the support. Initially, the support is given to a subset of A.

3 Programming ACs with RESAC

In this section, we shift focus to our resolution-based artificial chemistry, RESAC, which use concepts of FOPC to define the set of rules R and the set of objects S. We start the construction of RESAC, however, by defining the third component, the AC algorithm: The algorithm A driving the reactor M models a well-stirred reaction vessel (see Fig. 2). Thus arbitrary molecules can collide at any time. What seems to be an artificial simplification is, however, a concept of chemistry [14] if parameters like, for example, speed of stirring are chosen appropriately. Additionally, new start-clauses can feed the reactor—the so-called *inflow*—at a certain rate (*inflow rate*). The size of the vessel is kept constant, i.e. the inflow equals the dilution flux. An inflow rate $i, 0 \leq i \leq 1$,[2] indicates that $i * |M|$ start-clauses are inserted within one generation which, as usual, is defined to be $|M|$ collisions (independently of whether a collision causes a reaction or not). There are two special modes of inflow. First, an inflow rate 0 samples a closed reactor. Second, if a molecule is allowed to enter every time a reaction is not carried out (see below), we speak of an *elastic* inflow.

[1] Unification is the operation which is applied to terms in order to match them.
[2] synonymously referred to as $(i * 100)\%$-inflow

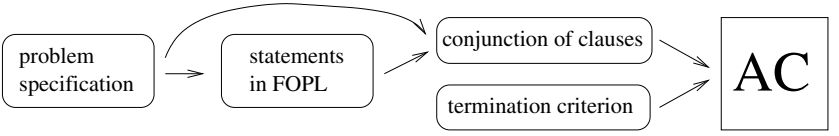


Fig. 3. Conversion steps from the original problem specification to AC

The set of objects S in the reactor covers molecules, each representing a clause of limited length[3]. These are arranged in a multiset-like structure [5] as implementation of the reactor model M. Thus all reactions performed map to set transformations. Input to RESAC are clauses $c_1, \ldots, c_n$, called *start-clauses*, which are derived from the problem specification (Fig. 3). Initially, the population holds η copies of each start-clause, and the size of the reactor $|M|$ equals $\eta * n$.

Concluding the description of RESAC, the remainder of this section is devoted to the reaction rule setup. The reaction scheme R between colliding molecules P and Q is given implicitly by the application of the resolution inference rule. If no resolution is feasible or permitted due to preconditions not met, the reaction is termed *elastic*, i.e. it does not take place at all. Otherwise, let V denote the inferred resolution result.This result is non-deterministic, because the pair of literals to be resolved is chosen randomly from the set of all resolvable literal pairs. Depending on a parameter *SELECT*, a reaction is then defined as follows:

$$\boxed{\textit{productive}\ \text{reaction:}}\ P + Q \Longrightarrow \begin{cases} U + V \text{ with } U \in \{P, Q\} & \text{if } \textit{SELECT=educt} \\ P + Q + V & \text{if } \textit{SELECT=free} \end{cases}$$

The parameter SELECT determines the implemented replacement policy. If *educt replacement* is applied the reaction scheme resembles an autocatalytic reaction with either molecule P or Q being preserved, whereas the other one is selected to be replaced by V. In contrast, *free replacement* lets V replace an arbitrary molecule.

The replacement policy matter is not discussed explicitly in most publications. Often free replacement is employed without comments on this issue. Although educt replacement may be considered closer to chemistry, it is rarely used (one exception is e.g. [7]). To reveal some of the differences between both replacement schemes, we investigate a basic setting with only three substances s_0, s_1, s_3, and simple reaction rule $s_1 + s_2 \Longrightarrow s_0$. Denoting the concentration of a molecule X at time step t with $|X|_t$, we set $(|s_1|_0, |s_2|_0, |s_0|_0) = (a, 1-a, 0)$, $0 \leq a \leq 1$. The following formulas are derived with $t \geq 0$, e stands for educt replacement, f stands for free replacement:

[3] In fact, the imposed restrictions are twofold: Molecules either too big or produced by too many reactions are not considered stable and are not allowed to be produced.

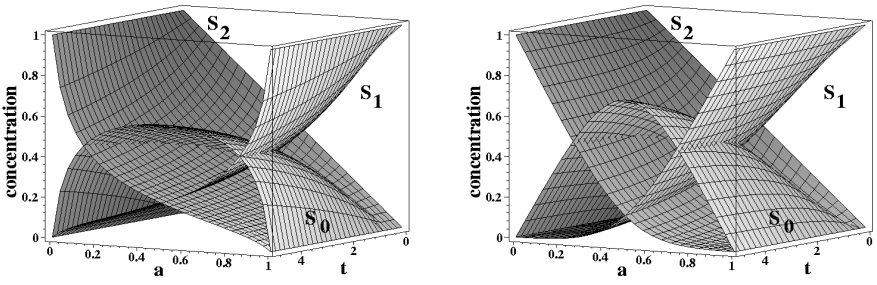


Fig. 4. Concentration transitions induced by $s_1 + s_2 \Longrightarrow s_0$ in a closed reactor. Explanations see text. *Left:* Free replacement. *Right:* Educt replacement

$$(|s_1|_t^f, |s_2|_t^f, |s_0|_t^f) = (\tfrac{a}{\sqrt{4\,(1-a)ta+1}}, \tfrac{1-a}{\sqrt{4\,(1-a)ta+1}}, \tfrac{\sqrt{4\,(1-a)ta+1}-1}{\sqrt{4\,(1-a)ta+1}})$$

$$(|s_1|_t^e, |s_2|_t^e, |s_0|_t^e) = \begin{cases} (\frac{a(-1+2\,a)e^{t(-1+2\,a)}}{-1+a+e^{t(-1+2\,a)}a}, -\frac{(-1+2\,a)(-1+a)}{-1+a+e^{t(-1+2\,a)}a}, 1-\sum_{i=1}^{2}|s_i|_t^e) & a \neq \frac{1}{2} \\ (\frac{1}{t+2}, \frac{1}{t+2}, \frac{t}{t+2}) & a = \frac{1}{2} \end{cases}$$

In Fig. 4 the concentration development is plotted. When educt replacement is employed, s_1 and s_2 are consumed completely in order to produce s_0 iff $|s_1|_0^e = |s_2|_0^e$. With free replacement, however, $\lim_{t\to\infty}|s_0|_t^f = 1$ holds for $0 \neq a \neq 1$. As expected, the global effect of free replacement leads to remarkably different behavior when leaving the center of the simplex given by (s_1, s_2). If $a = \frac{1}{2}$, both variants have the same limits but convergence speed differs. On this issue, in the next section empirical results of an example are stated.

4 Setup and Analysis – An Example

To demonstrate and analyze RESAC, we reconsider Adleman's famous experiment in which he initiated the cooperation of computer science and molecular biology for solving the NP-complete directed Hamiltonian path problem (DHPP) using DNA parallel processing [1]. Though not complex in the given size, this problem is suitable for our investigations because of its simple semantical structure.

Given a directed graph with designated vertices v_{start} and v_{end} the DHPP consists of finding a path (a sequence of edges) starting at v_{start} and ending up in v_{end} going through each remaining vertex exactly once. The graph shown in Fig. 5 depicts the graph G Adleman solved at the molecular level. To represent G in FOPC we have to transfer it to a number of clauses in conjunctive normal form. Using the function *edge()* and predicates *Path()* and *Visited()* we chose a tripartite system of clauses (see again Fig. 5). The first functional group is formed by the specification of the graph. Second, a "CONNECTOR" is introduced which

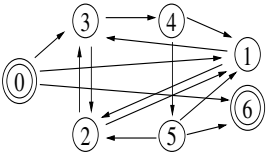
$(0 \to 1)$ Path(edge(0,1)), -Visited(0), -Visited(1) $(0 \to 6)$ Path(edge(0,6)), -Visited(0), -Visited(6) $\ddots$ $(5 \to 2)$ Path(edge(5,2)), -Visited(5), -Visited(2) $(5 \to 6)$ Path(edge(5,6)), -Visited(5), -Visited(6)	
(CONNECTOR) -Path(edge(X,Y)), -Path(edge(Y,Z)), Path(edge(edge(X,Y),Z))	
(SOLVER) -Path(edge(edge(edge(edge(edge(edge(0,X2),X3),X4),X5),X6),6))	
(TERMINATION) -Visited(0), -Visited(1), -Visited(2), ... , -Visited(5), -Visited(6)	
(SOLVER_{R1}) -Path(edge(0,X2)),-Path(edge(X2,X3)), ... , -Path(edge(X6,6))	
(SOLVER_{R2}) -Path(edge(edge(edge(edge(edge(edge(X1,X2),X3),X4),X5),X6),X7))	

Fig. 5. Representation of the Adleman problem in FOPC

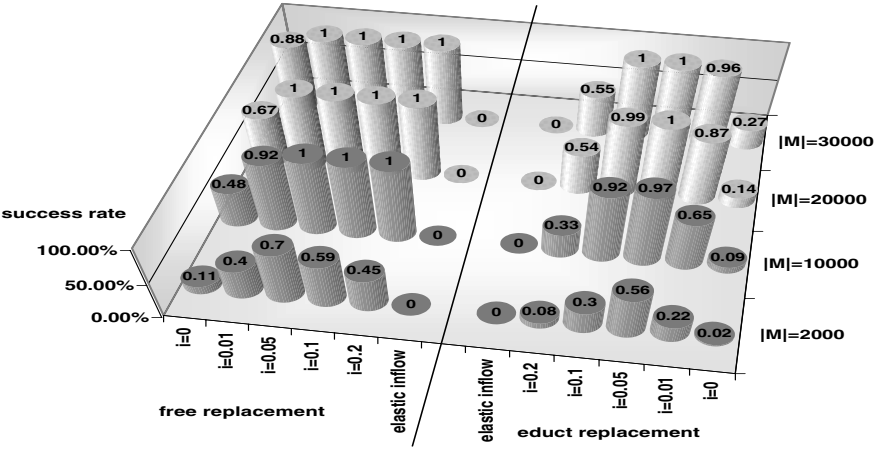
reflects the fact that if two paths $X \rightsquigarrow Y$ and $Y \rightsquigarrow Z$ are given also the path $X \rightsquigarrow Z$ exists. The solving clause "SOLVER" completes the definition of the set of start-clauses. Although the chosen representation is not optimal (see below) it is a somehow intuitive approach, and, moreover, it helps us to reveal some hidden properties of the different variants of RESAC. A simplified view of the solution process is as follows: The CONNECTOR joins pieces of paths to longer paths if possible. A path $v_{start} \rightsquigarrow v_{end}$ with 6 edges unifies and thereby reacts with the SOLVER. The *Path()*-literal is resolved, and a group of *Visited()*-literals remain. The terminating query criterion searches for such a group which comprises all nodes of G. If a reaction product matches this clause, a solution to this DHPP is found. By giving the support only to the SOLVER a kind of backward chaining is employed.

In our experiments we ran RESAC with 6 variants of inflow, 4 different sizes, and both replacement policies, each of these a 100 times. In Fig. 6 the percentage of solutions found (success rate), the average generation in which the solution was found (avg. success gen.), corresponding Std. Dev. and Std. Errors are listed. From the collected data we deduce 3 statements:

1. Free replacement outperforms or at least equals educt replacement in every experiment series with respect to a higher success rate, a lower avg. success gen., and a lower variability.
2. There seems to be an optimal inflow rate i_{opt}, such that for all inflow rates $i > i_{opt}$, it holds that the smaller i, the higher the success rate is, but, on the other hand, for inflow rates $i < i_{opt}$: the smaller i, the lower the success rate.
3. With increasing reactor size, the success rate is raising and solutions are found in earlier generations.

We now discuss these observations in indicated order:

(1): At first, we state the following two definitions which have been found useful for analyzing the system's dynamics: To measure the *productivity*, the number of productive reactions within one generation is divided by $|M|$. The *diversity* of M is the number of different molecules in M divided by $|M|$. In



	inflow rate	$\|M\| = 2000$	$\|M\| = 10000$			$\|M\| = 20000$			$\|M\| = 30000$		
free replacement	0	-	**-**	-	-	**-**	-	-	**-**	-	-
	0.01	-	**-**	-	-	**41.2**	*14.9*	1.49	**37.3**	*15.4*	1.54
	0.05	-	**40**	*16.5*	1.65	**30.3**	*11*	1.1	**26.4**	*8.4*	0.84
	0.1	-	**35.3**	*12.9*	1.29	**27.2**	*9.4*	0.94	**25.7**	*7*	0.7
	0.2	-	**39.8**	*17.5*	1.75	**30.4**	*10.3*	1.03	**26.1**	*7.8*	0.78
	elastic	-	**-**	-	-	**-**	-	-	**-**	-	-
educt replacement	0	-	**-**	-	-	**-**	-	-	**-**	-	-
	0.01	-	**-**	-	-	**-**	-	-	**-**	-	-
	0.05	-	**-**	-	-	**42.7**	*13.6*	1.36	**36.8**	*10.5*	1.05
	0.1	-	**-**	-	-	**-**	-	-	**39.1**	*12.8*	1.28
	0.2	-	**-**	-	-	**-**	-	-	**-**	-	-
	elastic	-	**-**	-	-	**-**	-	-	**-**	-	-

Fig. 6. Adleman experiment in various ACs: i denotes the inflow rate, $|M|$ the reactor size. Each experiment was repeated a 100 times. Run time was limited to at most 100 generations. In the graph the success rate is compared. A 5%-inflow shows the best results. In the Table the average success generation (typed in **bold**), the Std. Dev. (typed in *italic*) and the Std. Errors are listed; "-" indicates that not all experiments were successful

fact, diversity plays an important role explaining the first observation. Molecule diversity in reactors employing free replacement is significantly higher than with educt replacement (Fig. 8 shows examples). This yields a better distribution of solution candidates (clauses) within the search space.

If we look for the reason we have to go into detail. Listing the concentration development of all start-clauses in a closed reactor with educt replacement (Fig. 7) reveals the extinction of the CONNECTOR at an early stage of the experiment. As a consequence, productivity rate falls and approaches zero. What is the reason for this concise extinction of clauses which is a generally observed phenomenon?

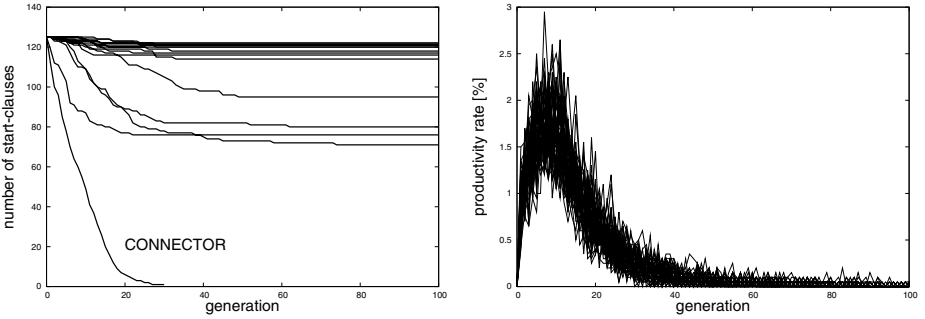


Fig. 7. Analysis of a closed reactor with size 2000 employing educt replacement. *Left:* Number of start-clauses. Note the development of the CONNECTOR. *Right:* Productivity in the same reactor[5]

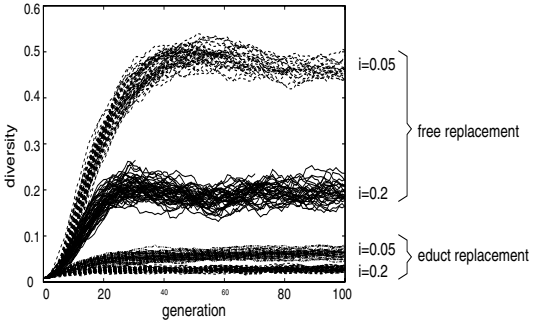


Fig. 8. Diversity in reactors with size 2000. Educt compared to free replacement. Diversity is significantly higher with free replacement. i denotes the inflow rate[4]

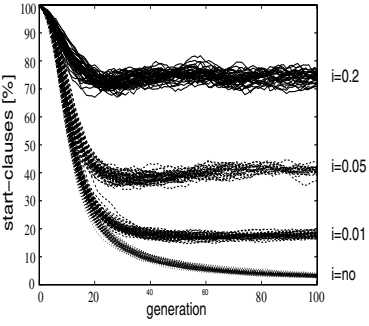


Fig. 9. A comparison of inflow variants i in a reactor with size 2000 employing free replacement[4]

With educt replacement one of the educts is replaced by the reaction product, thus, reactive molecules are more often replaced; a reaction pathway cannot be maintained if there is no inflow of the participating molecules. Moreover, diminishing molecules can prohibit the emergence of other pathways. On the other hand, non-reactive molecules obviously always are involved in elastic reactions and their concentration remains unchanged (niche building). Presumably useless most of the time, though, they use reactor space. As already seen in the basic experiment in the previous section free replacement policy, once again, has a global impact, thereby inducing a global pressure. But even if the CONNECTOR does not go extinct entirely, free replacement is superior because we observe a higher diversity of the free replacement reactor variants in open systems as well (Fig. 8).

(2): The higher the inflow, the higher is the fraction of start-clauses blocking the system by consuming valuable reactor space (Fig. 9). The elastic inflow

[4] A sample of 10 arbitrary runs is plotted simultaneously.

variant accompanied by immense inflow demonstrates this impressively (Fig. 10). In contrast, a low inflow rate leads to high diversity and a better covering of the search space. However, if the chosen inflow rate is too small, relevant start-clauses diminish which makes their interaction with other molecules a rare event, thus prolonging the solving process. To make, for example, a closed reactor successful, the run time and/or reactor size should be increased (see again Fig. 6). The optimal inflow rate i_{opt} is hard to find in the general case. In our experiments, a 5%-inflow has proven to be optimal.

(3): By enlarging the reactor, one can compensate for a smaller diversity caused, for example, by a high inflow rate. In generation 20 a reactor with 20000 molecules and 20%-inflow on average consists of nearly 2000 different molecules, approx. 3 times more than a 5%-inflow reactor with size 2000. In general, more molecules are (re)generated each generation (Fig. 11). Consequently, it is more likely to find a solution in an early stage of an experiment. Recalling the definition of a generation, however, this does not imply a faster execution. If measured in reactions, the large-sized reactor performs 10 times more reactions per generation than its small correspondent, such that a run is slower in terms of computational time.

Remark 1: Against the background of previous findings we may reconsider our representation of the Adleman problem. Basically, it has 2 drawbacks. The first difficulty is that "junk" clauses may emerge. Clauses considering more than 7 vertices are leading away from the goal. The second problem is the central focus on the CONNECTOR. Without restricting the solution space, one way to improve the situation is to remove this clause from the initial knowledge base and, instead, replace the SOLVER with SOLVER_{R1} (Fig. 5). In fact, results improve remarkably: With educt replacement, 10%-inflow and a reactor with 20100 molecules all 100 runs succeeded. The avg. success gen. is 5.94 (Std. Dev. 2.2). However, due to the implemented simplifications this artificially designed problem does not show relevant complexity necessary for representing a general problem.

Remark 2: A slight modification of the SOLVER lets us find all paths of length 6 (see SOLVER_{R2} in Fig. 5) . Every time the termination criteria is matched a directed Hamiltonian path is found. In this case, RESAC proceeds until a time limit is exceeded.

5 Conclusion

A computational system was set up whose overall behavior is characterized by emergent processes. With the resolution-based artificial chemistry *RESAC*, we intend to draw attention to the application-oriented branch of ACs, which probably is not as common as its use as a model ([10],p. 40). In previous publications the potential of resolution-based ACs was shown by implementing an automated theorem prover. In this paper, resolution logics enabled us to set up an AC based on a logic calculus. With this method, problems can be formulated within the AC framework by conversion to the first-order predicate calculus.

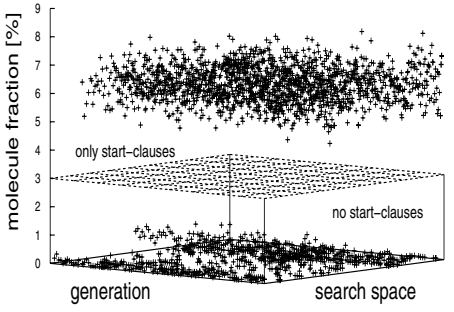


Fig. 10. Elastic inflow in a reactor with size 2000 (free replacement): The graph shows all clauses the system generates. All start-clauses have high concentrations—none of them is located below the plane. They block the system

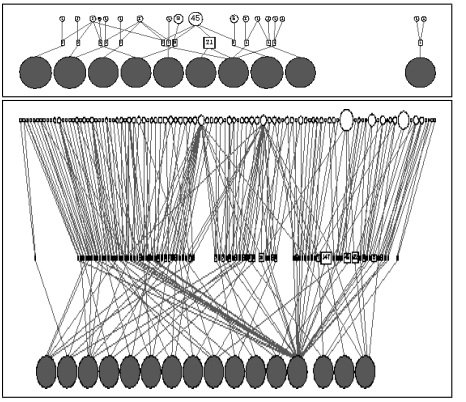


Fig. 11. Reaction network in generation 4, free replacement. Each line represents a reaction. *Top:* Reactor with 2000 molecules. *Bottom:* Reactor with 20000 molecules. Much more reaction pathways and molecules are built

We translated the directed Hamilton path problem presented by Adleman into FOPC. In 1994, Adleman implemented and solved the problem by means of DNA-computing, however, not much is revealed about the dynamics of such solving processes. Here, 48 variants of ACs were analyzed by investigating their dynamic behavior. Especially, the question how to proceed with the reaction product is addressed. Though these different setups cannot be very easily weighted against each other we identified some rules how performance of an application-oriented AC could be increased. These findings and the presented methods of dynamic system analysis may lead to a better understanding of the choices available at an early stage of AC design.

The presented system can easily be transferred to parallel computing platforms like multi-processor systems or distributed internet agents due to the inherent parallelism. Several realizations are currently under investigation.

Acknowledgments. We are grateful to Jens Ziegler for valuable comments and useful discussions.

References

1. L. M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.
2. J. C. Astor and C. Adami. A developmental model for the evolution of artificial neural networks. *Artif. Life*, 6(3):189–218, 2000.
3. R. J. Bagley and J. D. Farmer. Spontaneous emergence of a metabolism. In C. G. Langton, editor, *Artificial Life II, Proceedings*, Cambridge, MA, 1992. MIT Press.

4. R. J. Bagley, J. D. Farmer, and W. Fontana. Evolution of a metabolism. In C. G. Langton, editor, *Artificial Life II, Proceedings*, Cambridge, MA, 1992. MIT Press.
5. J.P. Banatre and D. Le Metayer. Programming by multiset transformation. *Communications of the ACM*, 36(1):98–111, 1993.
6. W. Banzhaf. Self-organisation in a system of binary strings. In R. Brooks and P. Maes, editors, *Proceedings of Artificial Life IV*, pages 109–118, Cambridge, 1994. MIT Press.
7. W. Banzhaf, P. Dittrich, and H. Rauhe. Emergent computation by catalytic reactions. *Nanotechnology*, 7:307–314, 1996.
8. R.A. Brooks. Coherent behavior from many adaptive processes. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Steward Wilson, editors, *From animals to animats 3*, pages 22–29, Cambridge, MA, 1994. MIT Press.
9. J. Busch. Automated Theorem Proving for first order predicate calculus using Artificial Chemistries. In GI Gesellschaft für Informatik, editor, *Informatiktage 1999*, Bad Schussenried, November 1999. Konradin Verlag Robert Kohlhammer GmbH.
10. P. Dittrich. *On Artificial Chemistries*. PhD thesis, University of Dortmund, Department of Computer Science, D-44221 Dortmund, Germany, January 2001.
11. P. Dittrich, J. Ziegler, and W. Banzhaf. Artificial Chemistries - a Review. *Artificial Life*, 7(3):225–275, 2001.
12. W. Fontana. Algorithmic chemistry. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II: Proceedings of the Second Artificial Life Workshop*, pages 159–209. Addison-Wesley, Reading MA, 1991.
13. W. Fontana and L. W. Buss. The barrier of objects: From dynamical systems to bounded organizations. In J. Casti and A. Karlqvist, editors, *Boundaries and Barriers*, pages 56–116. Addison-Wesley, 1996.
14. C.G. Hill. *An introduction to chemical engineering kinetics and reactor design*. John Wiley & Sons, 1977.
15. Y. Kanada and M. Hirokawa. Stochastic problem solving by local computation based on self-organization paradigm. In *27th Hawaii International Conference on System Science*, pages 82–91, 1994.
16. S. A. Kauffman. *The Origins of Order*. Oxford University Press, New York, 1993.
17. J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, January 1965.
18. T. Szuba and R. Stras. Parallel evolutionary computing with the random PROLOG processor. *Journal of Parallel and Distributed Computing*, 47(1):78–85, November 1997.
19. K.-P. Zauner and M. Conrad. Conformation-driven computing: Simulating the context-conformation-action loop. *Supramolecular Science*, 5:791–794, 1998.
20. J. Ziegler, P. Dittrich, and W. Banzhaf. Towards a metabolic robot control system. In M. Holcombe and R. Paton, editors, *Information Processing in Cells and Tissues*. Plenum Press, 1998.