Paper Category: Artificial Life

Chemical Crossover

Hugues Bersini
IRIDIA – Université Libre de Bruxelles
CP 194/6 - 50, av. Franklin Roosevelt
1050 Bruxelles – Belgium
bersini@ulb.ac.be

Abstract

During chemical reactions, molecules interact to produce new molecules. Some of the reaction mechanisms are very close to the way gene combine to produce new genes. Like the classical genetic crossover operator, chemical crossover occurs when two molecules exchange the atomic material they are composed of. Molecules do so in order to produce new and more stable molecules. Several differences exist however between the genetic crossover and its chemical version. Molecules are not coded as binary string but as computational tree instead. Moreover this computational tree, due to the symmetry in the way atoms are connected, must be organized in a strictly ordered way. Following a crossover, the resulting molecules must be reshaped according to precise organization rules. The crossover involves the exchange of single or multiple links. The fitness is distributed through the molecule such that only "better" molecules can result from the crossover. In this paper the chemical crossover and the computer simulation will be discussed within several perspectives: chemical, Alife and engineering. Simulation results will be presented for a simple chemical reactor composed of four atoms with different valence, and allowing molecules to deterministically or randomly interact according to the single-link-crossover.

1 INTRODUCTION

From its origins, Alife has always taken as fundamental the conception of software environments able to model the appearance and the disappearance of complexes constructed from simpler complexes or basic elements. This preoccupation leads to a succession of keystones in the Alife production such as Fontana's alchemy (1992), Kauffman's autocatalytic networks (1993) or Holland's echo models (1995). These software environments, like the major part of work being done (or which should be done) in the "Alife spirit", could have three major applications. Following an adequate parameterization, they could be used by chemical or biological practitioners who could find in these "computational platform" a helpful way to model a particular system. We can then speak of a set of computational design patterns, or software shell, which should easily be adapted to the simulation of a particular chemical or biological environment. On the other hand, these simulations can stand on their own and allow the discovery of generic laws, expressed in mathematics or linguistic terms, characterizing the behaviour of emergent and complex systems. For instance, in Kauffman's simulation of Boolean nets (1993), some laws establish the stability or instability of generic networks as a function of the range of their connectivity. Others set the number of attractors as a function of the number of units in the network. In our specific case, these laws could connect, for instance, the number and the maximal size of possible complexes with the number of basic elements, with their valence, the number of sources or any time constant for the concentration variation, etc. Finally, they can inspire the development of engineering practices which, by using the computational power in a simple but fully iterative and parallel way, can lead to very satisfactory solutions for highly complex problems.

This work departs from its ancestors by essentially relying on Object-Oriented (OO) computation instead of indifferent or ad-hoc computational tricks. Its originality

lies in the way the chemical environment is conceptualized in a network of classes and their specific interactions. From its origins, OO computation has allowed programming to come closer to physical simulation (the first OO language was indeed called "simula") instead of being constrained by the processor set of elementary instructions. Today there is a trend which make more and more possible to abstract software engineering from the processor by naturally using high-level natural concept making up the problems as the bricks of the resolution. This goes together with the increased use of visual modeling language such as UML. UML proposes a set of well defined diagrams (transcending any specific OO programming language) to naturally describe and resolve problems with the high level concepts inherent in the formulation of the problem. It is enough to discover the main actors of the problem and how they mutually relate and interact in time to build the algorithmic solution of this problem. It is beyond the scope of this paper to present UML although some of its symbols will be used to describe our chemical software environment. A simple and introductory overview of the UML language can be found in (Eriksson and Penker, 1998). However, by deliberately restricting our use of UML to the only class diagram, readers familiar enough with OO programming should not have any problem understanding this approach.

The second section of the paper will describe and discuss the basic classes of our chemical software environment, their main attributes and methods, and how they do relate to each other i.e. giving a verbal description of what compose the class diagram. The third section will describe in more details what is a molecule and how it is uniquely coded by a computational tree. In this work a molecule is neither a lambda-calculus expression (Fontana, 1992), any kind of operator (Dittrich, 1999; Dittrich and Banzhaf, 1998) nor a binary string (Farmer, Kauffman, and Packard, 1986; Holland, 1995), but is taken to be a computational tree, whose vertical and horizontal organization is strictly defined, due to the symmetry in the way atoms are connected. These organizational rules prevent two identical molecules, obtained as respective outcome of different reaction histories, to co-exist at any moment in the system. This is a replica, for our molecular computational tree, of the notions of "sameness" and "normal form" stressed by Walter Fontana is his lambda calculus framework (Fontana, 1992; Fontana and Buss, 1996). Those rules will be given and illustrated by numerous examples of molecules in section three.

In chemistry tutorials, it is frequent to find different names for different types of chemical reactions like: "decomposition" (when one molecule breaks down into two or more other simpler ones) or "combination" (when two molecules combine to form a new one) or "single and double replacement" (involving exchange of partners), etc. Based on the precise computational definition of a molecular identity given in the previous section, the section four and five will described in details one of these reaction mechanisms: the chemical crossover. The section will show how do the molecules exchange their link and how the products are reorganized. Section six will show the results of some simulations obtained, departing from four atoms of valence 1,2 and 4, and which compose four elementary molecules of two atoms. These four diatomic molecules will initiate a chain of reactions where two molecules picked randomly will be able to interact according to the single-link-crossover reaction. The last section will show what happens to the simulation when we allow, besides the metadynamics (leading to an artificial chemistry said "strongly constructive" (Fontana, 1992; Dittrich, 1999)), the molecule to change their concentration in time. A simple first-order reaction will be simulated. Various algorithmic alternatives for the whole simulation loop will be discussed. Whereas in a lot of Alife simulations, the dynamics and the metadynamics are kept separated, a clear interest of this work is their simultaneous consideration.

2 THE OO CHEMISTRY CLASS DIAGRAM

A lot of similarities are found with the UML class diagram presented in an earlier paper (Bersini, 1999). The **Component** super class includes as main attributes the *concentration*. This concentration changes in time by natural decrease or increase and as a result of the reactions and their specific rate, in a way that will be discussed in section five.

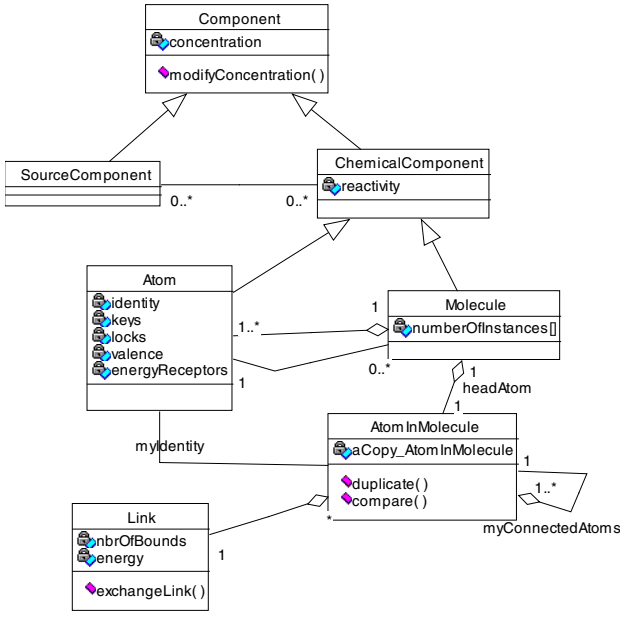

Figure 1: The UML class diagram of the OO Chemistry

The **Chemical Component** is a subclass of Component and the super-class of two further sub-classes: Atoms and Molecules. The **Atom** is the first sub-class of the Chemical Component and describes the basic objects of the whole system. The fundamental attribute is the *valence*, which indicates in which proportion this atom will connect with another one to form a molecule. For instance an atom with valence 4 (for instance with identity "1" for reasons to appear later) will connect with four atoms of valence 1 (for instance with identity "4") to form the molecule: *1(4 4 4 4*). Connections between atoms are perfectly symmetric.

The second major attribute is the *identity*, which, in our simulation, relates to the value of the variance. Atom with a high variance will be given a small identity index. This identity simply needs to be an ordered index ("1", "2", …) for the organization rules shaping the molecular tree be possible. The way this identity is defined depends on what we take to be unique to any atom. In chemistry this identity is given by the atomic mass i.e. the number of protons and neutrons.

Molecule is the second sub-class of Chemical Component. The following section will show how are the molecules structured in a unique way. As shown in the class diagram, molecules are compounds of atoms. An attribute called *numberOfInstances* is a vector of integers whose elements are the number of times one specific atom appears in the molecule (i.e. four "4" and one "1" in the molecule *1(4 4 4 4)*). Molecules are trees that are computationaly structured with pointers of class **AtomInMolecule**. Each molecule possesses one and only one AtomInMolecule pointer called the *headAtom* and which can be seen as its "front door" (it would be the "1" in the molecule *1(4 4 4 4)).* As soon as an atom enters into a molecule, it is transformed into an AtomInMolecule object. AtomInMolecule relates to atom since the identity of such an object is the same as its associated atom. AtomInMolecule are responsible for coding the tree structure (Aho and Ullman, 1995) of the molecule since they possess pointer attributes (called *myConnectedAtoms*) pointing to a vector of AtomInMolecule objects.

An addition with respect to the previous work is the class **Link**. An object "Link" connects two AtomInMolecule. It has a given *energy* so that the weakest link is the first to break, and a *number of bonds*. For instance two atoms of valence 4 will connect (to form a diatomic molecule *4(4)*) with a link containing 4 bonds, and one atom of valence 4 will connect with four atoms of valence 1 (*1(4 4 4 4)*), each link containing one bond. Link objects intervene in the unfolding and the coding of the reaction mechanisms. For instance, one major method associated with the class Link is "*exchangeLink"* involved in crossover molecular reactions.

3 THE MOLECULAR UNIQUE COMPUTATIONAL STRUCTURE

For facility and space, the following linear notation will be adopted to describe a molecular computational tree. One example will be enough to understand it. Take the following molecule:

1 (1 (4 4 4) 2 (1 (3 3 3)) 2 (2 (3)) 2 (4))

"1" is an atom with valence 4, "2" is an atom with valence 2, and "3" and "4" are atoms with valence 1. The graphic tree version is given in figure 2.

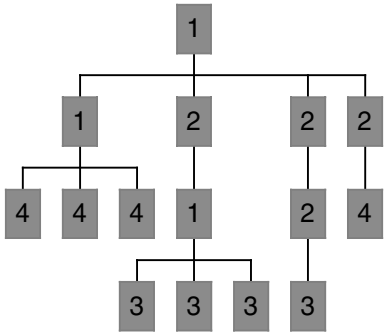

Figure 2: The molecular graphic tree

The following rules need to be respected in order to shape the molecular tree in a unique way:

Vertically: The highest node, i.e. the front door of the molecule (the initial "1" in our example of fig.2) must be the smallest of all the AtomInMolecule objects composing the tree.

Horizontally: Below any node (i.e. any AtomInMolecule) the sub-nodes are arranged from left to right in an increasing order, the smallest to the left, the greatest to the right.

Clearly these two rules depend on the definition of "smaller" between two AtomInMolecule nodes. It is defined in a way described in table 1 for two AtomInMolecule "n" and "m". You can see that the example given in fig.2 indeed respect these rules: the highest "1" is connected first to a "1" then to a "2" whereas the other "1", although first connected to a "1", are afterwards connected to atoms with higher identity. The horizontal rule is equally verified for all connected atoms.

if (the identity of n < the identity of m) {the smaller is n }
else
if (the identity of n = the identity of m)
{if (n has no connected atom and m has no connected atom) {the smaller is n}
else
if (the number of connected atoms of n > the number of connected atoms of m) {the smaller is n}
else
if (the number of connected atoms of n = the number of connected atoms of m)
{for all j connected atoms of n and m
{if (the identity of the jth connected atom of n < the identity of the jth connected of m) {the smaller is n , break-the-loop}
else
{ for all connected atoms of n and m
{ redo recursively the same testing procedure}}}

Table 1: Which is the smaller between the atomInMolecule n and m.

4 THE CHEMICAL CROSSOVER

Several reaction mechanisms called "decomposition", "combination", "replacement" are repeatedly described in the chemical literature. Based on our syntactical definition of what is the molecular identity, we will here only focus on one common type of chemical reactions called the chemical crossover, with two instances: the single-link and the multiple-links. Every time a new molecule is created as a result of the combination of two molecular reactants, this new molecule needs to be reshaped according to the organization rules previously defined (transformed into its normal form (if borrowing Fontana's words (1996)).

- *Single-link crossover*: the weakest links of each molecule are exchanged (remember that "1" has valence 4, "2" has valence 2 and "3" and "4" have valence 1, and suppose the link "2-3" is weaker than the link "2-4").

  *1(1) + **[4]** 2 (3 4) → 1(3 3 3 3) + 1(2(4) 2(4) 2(4) 2(4))*

  The bold values between brackets are stoichiometric coefficients needed in order to balance the chemical equations. Famous chemical reactions like for instance the neutralization reaction are typical single-link crossover, for instance:

  $$HCl + NaOH \rightarrow NaCl + H_2O$$

- *Multiple-link crossover*: several weak links are homogeneously exchanged between the two molecules.

  *1(4 4 4 4) + **[2]**2(2) → 1(2 2) + **[2]**2(4 4)*

  The linear notation cannot account for the number of bonds characterizing the links. Here the 2-2 and the 1-2 links are double bonds.

  $$CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O$$

5 THE DIFFERENCES WITH THE GENETIC CROSSOVER

Since the essential similarity between GA and the chemical simulation presented here lies in the recombination operator, it is worth discussing in details the differences existing between the two operators and the engineering consequences of these differences.

1. Crossover occurs between computational trees instead of binary strings, and by links exchange (see fig.3). The recombination of individuals coded in a different way than binary string has become very common in the GA community. Indeed recombining computational trees is at the basis of genetic programming (Koza, 1992). Crossover reaction between molecules is more akin to lisp automated programming than simple GA.

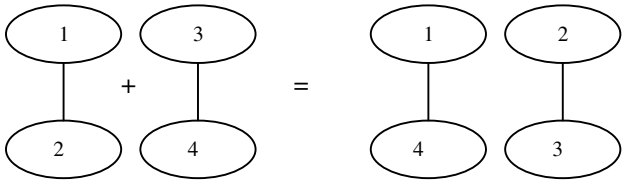

Figure 3: a single-link-crossover

2. Chemical valence plays the following role. If a link containing n bonds exchange with a link containing m bonds, the new individuals will be connected by n/m links:

   *1(1) + **[4]**3(4) → 1(3 3 3 3) + 1(4 4 4 4)*

   There is no engineering application that could positively benefit from this chemical fact.

3. Like seen previously, the crossover can involve more than one links in one of the two molecules, to become a multiple-links crossover. This type of crossover can equally be found in engineering applications where the repetition of identical schema to be exchanged at one go is not impossible.

4. A major aspect in our simulation is the reshaping of the molecular trees in a strictly specified order after the recombination has occurred. In figure 3, the second molecule obtained after the crossover is not 3(2) but 2(3) to respect the vertical ordering rule. No practice of genetic algorithm raises this problem of re-transforming the obtained individual in a canonical form. Every obtained individual exists in its resulting form. However engineering problems where the solutions present important symmetry could be found, requiring their subsequent reshaping to avoid repetition and redundancy. Binary string and lisp codes have no problem of symmetry but trees or more sophisticated computational structure could have.

5. A final aspect is the way fitness and selection enter into the simulation. In real chemistry only reaction that lower the energy (assimilated here with a fitness increase) of the products as compared with the energy of the reactants are possible. In fact the fitness increase is a basic condition for the reaction to occur. It is akin to a form of hill-climbing where only improvement is possible. As a matter of fact molecules are structured in such a way that it is easy to pinpoint the material to exchange (here the links) in order to increase the fitness or decrease the energy. Moreover molecule seem to present weak epistatic interactions because the contribution of the links to the global energy of the molecule can be individuated. With respect to classical GA, this could drive to algorithmic versions where the fitness is distributed among the schema and whenever a recombination occur, only the schema contributing the most to the fitness would be recombined. Notice that this type of recombination exists and has been already discussed in the GA community (Baluja and Caruana, 1995). Obviously the weaker the epistasis the more meaningful this type of recombination appear.

6 SIMULATING THE SINGLE-LINK CROSSOVER

We now show the results obtained by running the chemical simulators with the four atoms “1” (valence 4), “2” (valence 2), “3” (valence 1), “4” (valence 1) and the four basic diatomic molecules: *1(1), 2(2), 3(3), 4(4).*

The simulator runs in the following way:

1. Take randomly two molecules
2. Make them interacting according to the single-link-crossover. The link to exchange in each molecule is the weakest link (each link receives a random energetic value).
3. Generate the new molecules in their organized form only if they do not already exist in the system.

A list of some firstly obtained molecules is given below:

1 (2 2) , 1 (3 3 3 3) , 2 (3 3) , 1 (4 4 4 4) , 1 (1 (2) 1 (2)) , 1 (2 3 3) , 1 (2 4 4) , 1 (1 (2) 2) , 1 (1 (3 3 3) 1 (3 3 3) 1 (3 3 3) 1 (3 3 3)) , 1 (2 (1 (3 3 3)) 3 3 3), 1 (1 (1 (3 3 3) 2) 3 3 3), 1 (2 (3) 2 (3) 2 (3) 2 (3)) ,1 (2 (3) 2 (3) 2) , 1 (2 (3) 3 3 3) , 1 (1 (4 4 4) 1 (4 4 4) 1 (4 4 4) 1 (4 4 4)) , 1 (2 (1 (4 4 4)) 4 4 4) , 1 (1 (1 (4 4 4) 2) 4 4 4) , 1 (1 (4 4 4) 4 4 4) , 1 (2 (4) 2 (4) 2 (4) 2 (4)) , 1 (2 (4) 3 3 3) , 1 (2 (4) 4 4 4) , 1 (1 (1 (2)) 1 (1 (2))) , 1 (1 (2) 3 3) , 1 (1 (2) 4 4) , 1 (1 (1 (3 3 3) 1 (2)) 3 3 3) , 1 (1 (1 (4 4 4) 1 (2)) 4 4 4)

It is worth verifying the correct vertical and horizontal organization of all the molecular trees to see the rules presented in the section three in action.

After several seconds of simulation you easily get molecules long of 53 atoms like the following one:

1 (1 (1 (3 3 3) 1 (3 3 3) 1 (3 3 3)) 1 (1 (3 3 3) 1 (3 3 3) 1 (3 3 3)) 1 (1 (3 3 3) 1 (3 3 3) 1 (3 3 3)) 1 (1 (3 3 3) 1 (3 3 3) 1 (3 3 3)))

or 101 atoms like:

1 (1 (1 (3 3 3) 1 (3 3 3) 1 (1 (1 (3 3 3) 1 (3 3 3) 1 (3 3 3)) 1 (2))) 1 (1 (3 3 3) 1 (3 3 3) 1 (1 (1 (3 3 3) 1 (3 3 3) 1 (3 3 3)) 1 (2))) 1 (1 (3 3 3) 1 (3 3 3) 1 (1 (1 (3 3 3) 1 (3 3 3) 1 (3 3 3)) 1 (2))) 1 (1 (3 3 3) 1 (3 3 3) 1 (1 (1 (3 3 3) 1 (3 3 3) 1 (3 3 3)) 1 (2))))

You could substitute “1” with “C”, “2” with “O” and “3” with “H” in order to obtain realistic organic molecules.

7 THE DYNAMICS COMES INTO PLAY

Every chemical component, and thus molecule, is assigned an important attribute: its concentration (designated by “conc()” in the following) which can change in time. In the next simulations to be shown, like for classical chemical kinetics, the molecular concentration will change as the sole effect of the

reactions. For instance in a first-order reaction whenever an interaction takes place between two molecules A and B to give two new products E and F:

$$A + B \rightarrow E + F$$

conc (E) = conc (F) ← k*conc(A)*conc(B)

if E and F don't exist already. *k* is called the reaction rate.

conc (E) ← conc (E) + k*conc(A)*conc(B)

if E already exists (the same for "F")

And in all cases:

conc(A) ← conc(A) – k*conc(A)*conc(B)
conc(B) ← conc(B) – k*conc(A)*conc(B)

Take the elementary case of a chemical reaction involving only two one-valence atoms "3" and "4" composing two diatomic molecules 3(3) and 4(4), and interacting by single-link-crossover:

3(3) + 4(4) → **[2]** 3(4) (the [2] is the stoichiometric coefficient)

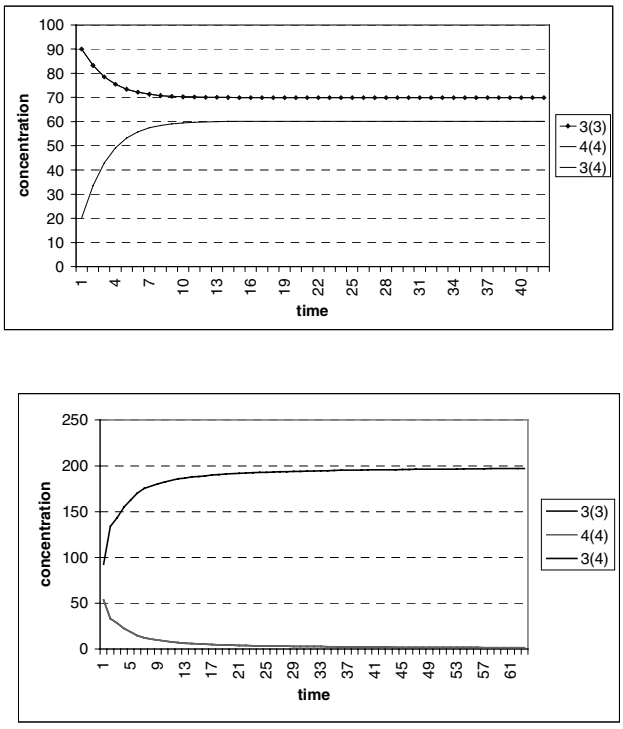

Figure 4: Time evolution of the concentration of the three interacting molecules: up – in the reversible case, down – in the irreversible case.

In figure 4, you can see two plots showing the concentration evolution in time of the three molecules 3(3), 4(4) and 3(4): The first plot in the case of a reversible reaction, the second one in the case of an irreversible reaction.

These two plots experimentally obtained could be analytically obtained by resolving in this case the simple differential equations. While the analytical road is possible for very simple reaction mechanisms, it is far to be the case for much more complicated reaction schemes involving a lot of intermediaries. For these reactions, rapidly leading to a formidable analytical complexity, one is forced to resort to computer simulations. The complete simulation of our chemical systems can take three forms. The first is *random interaction* shown in table 2.

Ad infinitum do {
- time = time + 1
- Select randomly one molecule A
- Select randomly one molecule B
- Make the reaction (A,B) according to a specific reaction scheme
- If the products of the reaction already exists, increase their concentration, if not add them in the system with their specific concentration.
- Decrease the concentration of A and B
}

Table 2: The random interaction algorithm

In the case of several interacting molecules, figure 5 shows the evolution in time of 5 molecules among others.

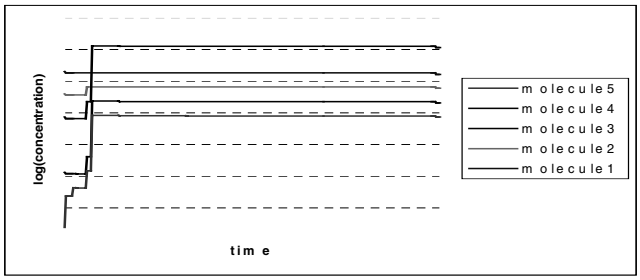

Figure 5: Time evolution of the concentration of the interacting molecules for the random interaction algorithm.

Now the random interactions, although commonly found in a lot of artificial chemistry schemes (Fontana, 1992; Dittrich, 1999) does not make a lot of sense if the molecules are not single chemical objects but are concentration of chemical objects. In this case, the reaction rate "k", to some extent, already account for the randomness of the molecular collisions. It becomes more natural to resort to a deterministic type of simulation such as indicated in table 3.

```
Ad infinitum do {
- time = time + 1
- For all molecules i of the system
   For the molecules j  ( = 1 to i) of the system
   { -  Make the reaction (i,j) according to a specific
  reaction mechanism
     - If the products of the reaction already exist, increase
their concentration, if not add them in the system with
their     specific concentration.
     -  Decrease the concentration of i and j }}
```

Table 3: The deterministic interaction algorithm

Although the best algorithmic version, the problem with such an algorithm, is that everything rapidly exponentially explodes so that after 4 or 5 time increments, the simulation is captured in a nearly infinite loop (at time t=0 : 10 reactions, at t=1: 55 reactions , at t=2:1540 reactions etc.).

A final algorithmic compromise is still deterministic but takes the time increment inside the double loop such that a reaction occurs at every time step, with nevertheless the sequence of molecular interaction kept fixed. The time evolution of some molecules is shown in figure 6.

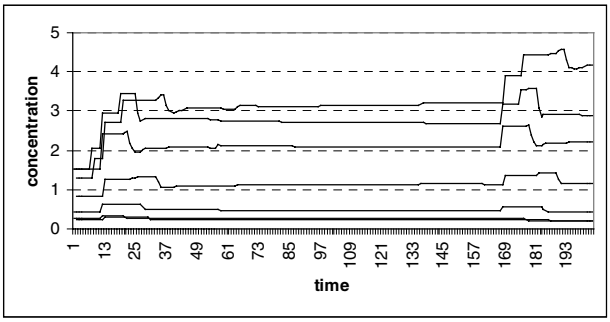

Figure 6: Time evolution of the concentration of seven molecules for the deterministic interaction algorithm

Interestingly enough, this version of the algorithm allows the detection of chaining reaction (which is related with the Fontana's level 1 of self-maintaining molecular system (1992)). As a matter of fact, the concentration of certain molecules seems to oscillate in time. This is only possible if some reaction intermediaries are first consumed then regenerated as products of subsequent reactions. Fully self-maintaining molecular loops in which all molecules are the products of some interaction involving other molecules can be distinguished from weaker chaining reaction in which only some of the molecules are obtained as a consequence of reactions.

However the observed cycles are nothing but artifactual effect of the version of the algorithm. They would naturally disappear to give fixed point in the more exact version of the algorithm (version 2). Indeed, the figure 4, in the reversible case, the smallest version of a molecular self-maintaining system, shows clearly the appearance of fixed point. This computer artifact reminds a similar effect and related discussion concerning "the asynchronous versus synchronous updating" found in the cellular automata literature (Bersini and Detours, 1994).

So while chaining reactions and self-maintaining molecular systems are certainly common in the whole reactor, their precise detection could be harder among the multitude of fixed points that would allow the simulation of the whole system, provided much computing power is available.

8 CONCLUSIONS

The kind of computational chemistry being investigated in this work lies at an abstraction level that might allow the simultaneous fulfillment of two of the main objectives of Alife. A first one is to offer chemists a computational platform, which following an adequate parameterization, could help them to model and understand the evolution of a particular chemical system. Indeed a large number of key aspects remain to be explored and tuned in the simulation among which:

- the value of the reaction rates
- whenever two molecules collide, which reaction mechanism takes place and which links in each molecule are involved in this reaction
- the kind of relationship that exists between the reaction rates and the composition of the molecules involved in the reaction.
- the dependence of the reactions on the energy sources

Getting all this missing knowledge, it might be possible to simulate in a distant future the historical Miller's experiment aiming at replicating the origin of life. This could shed some new light on the length of the organic molecules obtained by Miller, of surprising complexity but still so far from the complexity of a DNA.

On the other hand, Alife simulations can stand on their own and lead to the discovery of generic laws characterizing the behavior of complex systems. When concerned with chemistry, the best representative of this research road is Walter Fontana (1992, 1996) who is trying to develop a pure mathematical abstraction allowing to better formalize the appearance of interesting dynamic and self-maintaining structures. His assimilation of molecules with operators of the lambda calculus, and reactions with molecules operating one on another seems to be unrelated with this work.

However, it is possible that the algorithmic choices made in our simulation (molecules = computational trees and reactions = symmetric combinatorial exchanges or combinatorial re-organization of the trees), closer to real chemistry, could still appear as a possible instance of Fontana's mathematical developments. If this is the case, we could have for the same price a chemical computational platform benefiting from his mathematical solid progress while able to present a friendly interface to chemical practitioners.

Finally in a more engineering perspective, chemistry seems to show aspects already found in various GA or GP applications like the computational trees recombination or the distribution of the fitness on the schema (in our case the links) to be recombined. The reshaping of the obtained individuals after crossover could appear in engineering applications where the solutions, like molecules, present strong symmetry in their organization.

References

1. Aho V. A., Ullman, J.D. (1995): Foundations of Computer Science – Computer Science Press - W.H. Freeman and Company – New York.
2. Baluja, S. and R. Caruana (1995). Removing the Genetics From the Standard Genetic Algorithm. In Proceedings of the Twefth International Conference on Machine Learning, ML-95, Edited by A. Prieditis and S. Russel, pp. 38-46. Palo Alto, CA: Morgan Kaufman.
3. Bersini, H. (1999): Design Patterns for an Object-Oriented Computational Chemistry – In proceedings of the 5th European Conference on Artificial Life (ECAL'99) – Eds : Floreano, Nicoud, Mondada – Springer – Verlag - pp. 389-398
4. Bersini, H. and V. Detours: Asynchrony induces stability in cellular automata based models – In proceedings of Artificial Life V – Eds : Brooks and Maes – MIT Press – pp. 382 – 387.
5. Detours, V., Bersini, H., Stewart, J. and Varela, F. (1994): Development of an Idiotypic Network in Shape Space – Journal of Theor. Biol. (170), 401-404 (1994)
6. Dittrich, P. and Banzhaf, W. (1998): Self-evolution in a constructive binary string system. Artificial Life, 4(2):203-220.
7. Dittrich, P. (1999): Artificial Chemistries – Tutorial held at ECAL'99 – European Conference on Artificial Life 13-17 September, 1999 – Lausanne, CH.
8. Eriksson, H-E, Penker, M. (1998): UML Toolkit – John Wiley and Sons
9. Farmer, J.D., Kauffman, S.A. and Packard, N.H. (1986) Autocatalytic reaction of polymers. Physica D, 22:50-67.
10. Fontana, W. (1992): Algorithmic Chemistry. In Artificial Life II: A Proceedings Volume in the SFI Studies in the Sciences of Complexity (C.G. Langton, J.D. Farmer, S. Rasmussen, C. Taylor, eds.), vol. 10. Addison-Wesley, Reading, Mass (1992)
11. Fontana, W. and L.W. Buss (1996). The barrier of objects: From dynamical systems to bonded organization in Casti, J. and Karlqvist, A., editors, Bondaries and Barriers, pages 56-116. Addison-Wesley.
12. Holland, J.H. (1995): Hidden Order – How adaptation builds complexity – Helix Books – Addison Wesley Publishing Company (1995)
13. Kauffman, S. (1993): The Origins of Order: Self-Organization and Selection in Evolution – Oxford University Press
14. Koza, J.R. (1992) Genetic Programming. On the Programming of Computers by means of Natural Selection. Cambridge, MA MIT Press