Reaction Mechanisms in the OO Chemistry

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

Abstract

The work presented in this paper continues the work presented in the previous Alife (Bersini, 1999) conference by improving the connections between the computerized object-oriented chemistry and the real natural chemistry. This improvement is the result of a clearer definition of the type of computational structure which codes for a molecule, of the different reaction mechanisms between two molecules to obtain one or several new ones, and of the way the dynamics (i.e. the molecular concentration change in time) and the metadynamics (i.e. the appearance of new molecules) simultaneously make the whole chemical system to evolve in time. What makes a molecule computationally unique, in terms of a strictly ordered tree, will be deeply described. Simulation results are presented for a simple chemical reactor composed of four atoms with different valence, and allowing molecules to deterministically or randomly interact according to two mechanisms: the chemical single-link-crossover and the open-bond reaction. How the chemical kinetics is called into play will be shown for very elementary reactions.

Introduction

An "existential problem" inherent in a lot of works being developed under the Alife banner is their exact positioning between the natural sciences from which they originate: biology or chemistry, and some very abstract scheme, appealing at a pure computational level, but too much abstract for the results obtained to be of any feedback on the inspiring natural sciences. This position is so unstable that the temptation is great to escape this neutral situation by joining one of the two extremes, either to become a pure natural scientist manipulating test-tubes, or just dealing with software in a close formal or engineering perspective. As a consequence of this instability, this work attempts to slightly fill in the gap between the computer simulations of the chemical components and the way they interact on one side and their natural counterpart on the other side.

Physics naturally strides towards the "dematerialization" of objects by restricting their study to the laws of behavior and the laws of interaction. The "physical nature" of an objet reduces to a set of variables, the measure of which is experimentally possible, and that can easily be manipulated in a computer. Chemistry on the other hand fully integrates material object in its study, and the challenge is great for whatever computational approach, proposed in Alife, to see how the borders of materiality can be crossed. Several times in this paper, this difficulty will recur. While computational biology is similarly confronted with the limits of materiality, it might be judicious to attack this limitation one level below by allowing the capture of a reality with is known to be easier to model. That is while chemistry appears to be a key target for attempts of computational replica, and entails artificial chemistry to strategically precede artificial biology.

No doubt that we are today very far to be able to replicate in software and in all details a real molecule and the simplest interaction between two of them. It turns out to be pretty laborious to integrate in whatever computer simulation capital aspects as the 3-D shape, the energetic and electronic properties, both of a single molecule and of the way they interact. What is possible however is to capture in software, and in a preliminary attempt, some basic scholar chemistry like the molecular composition and combinatorial structure, basic reaction mechanisms as chemical crossover or bonds opening/closing and simple dynamical aspect as first or second order reactions.

A lot of chemical aspects is still however left aside (which reaction takes place, which link is involved, what is the value of the reaction rate,...) but in such a way that it should be easy for a more informed chemist to parameterize the simulation in order to account for the influence of these aspects.

The choice of object-oriented (OO) programming naturally follows from this attempt to reduce the gap between the computer simulations and its natural counterpart. It is intrinsic to OO programming that by thinking about the problem in real-world terms, you naturally discover the computational objects and the way they interact. In contrast, procedural programming forces the conversion of the real-world problem into the computer basic language. Like in our precedent paper, the next section will recall the UML class diagram, recapitulating the seven main classes of our chemical software and the way they relate to each other. UML diagrams (Eriksson and Penker, 1998) allow understanding the central structure of the program, and the interaction of its modules, without resorting to the code. As a matter of fact, the section will plead for an increased use, in our community, of tools of analysis like UML. UML eases the development of the

model but, first of all, helps the communication of this model with no need to get into a lot of implementation details.

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 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. Also the section will precise the kind of chemical isomers that can be differentiated when represented as computational tree, and the isomers that this representation can't allow to differentiate.

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 three will propose a new nomenclature for several reaction mechanisms, each of them being precisely described: the single-link-crossover, the multiple-links-crossover, the open-bond and closed-bond reactions, the changing-variance and the re-organizational reactions. Section four 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 only two of the reactions: the single-link-crossover and the open-bond 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. Artefactual effects resulting from different algorithmic choices will be illustrated and 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.

The OO Chemistry Class Diagram

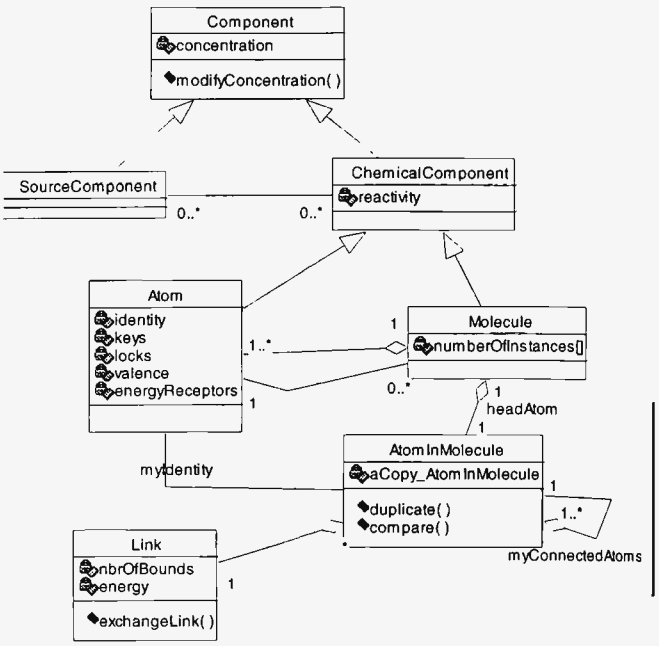


Figure 1: The UML class diagram of the OO Chemistry

It is not OO programming language (java, c++, smalltalk,) whose use is advocated here (although java contributes a lot to the spreading of Alife applets through the Internet). Everything programmed in an OO language can be alternatively done in a procedural way. It is rather the exploitation for communication ends of what OO languages, by increasing their use, have made more and more necessary, useful and prevalent in the computer world. That is, the development of high-level graphical notations to describe what the program is doing and how the reality is actually simulated without the need to resort to the code. The Unified Modeling Language (UML) is a major opportunity for our community to improve the deployment, the better diffusion and better understanding of the Alife models, and especially when addressed to researchers not feeling comfortable reading programs.

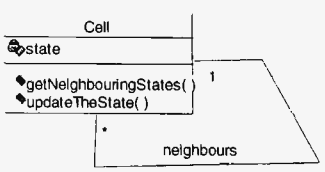


Figure 2: The Cellular Automata UML class diagram

Take the class diagram whose main function is to identify the classes and how they do interact, and see in fig.2 the simplest example of what a cellular automata basically reduces to. Each cell interacts with n neighbors by just getting their state and, on the basis of this value, updating their own state.

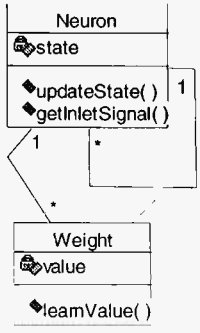


Figure 3: The UML neural net class diagram

A neural net (fig 3) has an additional class, the "Weight" that can learn its value. One neuron is associated with n weights and n neurons, since updating the state of a neuron needs the value of the connected weights and neurons. Additionally, each weight links together two neurons. Class diagrams are eventually more useful in simulations involving a large number of classes. A typical example is ecosystem simulations that generally contain different sorts of predators, preys, and resources (with neurons in these creatures and sometimes genes...). On the other hand, sequence diagrams can be useful when it is the sequence of events that counts to identify the process. For instance, a critical debate, very vivid in our community, addresses the choice of synchronous or asynchronous updating in CA kind of models. A sequence diagram easily helps to pinpoint the difference between these two updating mechanisms.

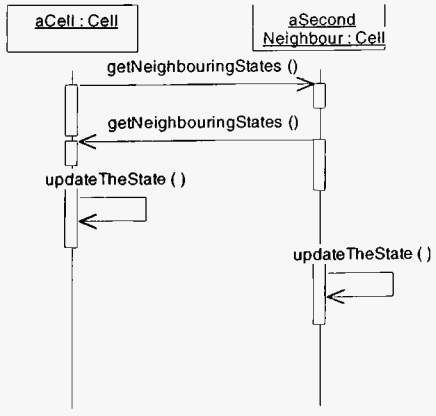


Figure 4: A UML sequence diagram helping to differentiate synchronous from asynchronous updating

Fig.4 represents the sequence diagram in the synchronous case for two cells of the cellular automata. For the asynchronous case, it is enough to shift the second "getNeighbourState()" message below the first "updateState()".

In the same line of thought, the use of UML is this paper aim at easily describing to computer scientists and chemists the components of the chemical simulation (fig.1) and how they do interact. 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. 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 value. 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. Actually, 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 the molecules are 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.

The Molecular Unique Computational Structure

Whatever chemical notation you adopt, for instance the "line-bond" or Kékulé structure (for instance the butane molecule in fig.5), one way of reproducing the connectivity pattern of a molecule is by means of a computational tree.

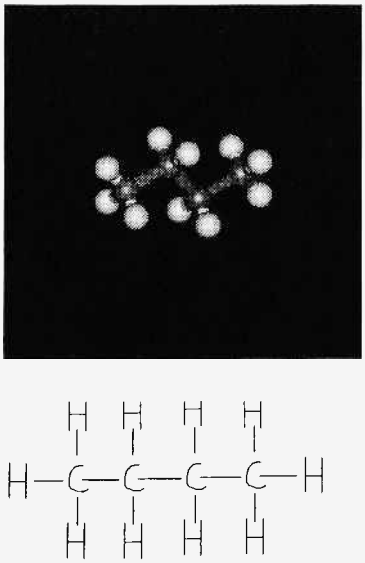


Figure 5: The "butane" molecule

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 fig.6. In our linear notation, the "butane" molecule of fig.5 would become:

C(C (C (H H H) H H) C (H H H) H H)

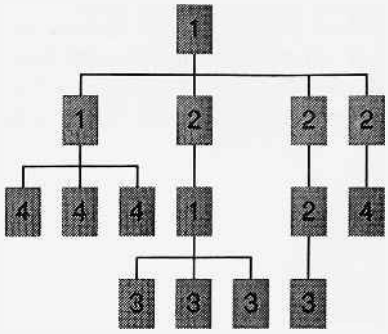


Figure 6: The computational tree structure for a molecule

```
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.

In a first approximation, the connectivity shows symmetry both vertically and horizontally. 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.6) 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.6 indeed comply with 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.

Organizing the tree is such a way allows differentiating two structural isomers, i.e. molecules that contain the same number of the same atoms but in a different arrangement (like the well-known chemical examples of the butane and the methylpropane molecules, both C_4H_{10} but with distinct connectivity patterns). However such a re-organisation can miss the differences still remaining between molecules showing a same connectivity pattern but with different spatial organisations i.e. the geometrical isomers. These different spatial organisations can induce different optical properties, which is of no relevance for the remaining of the paper, but can also play a major role in the type of reactions these molecules are subject of. This consists of a first major obstacle coming from the "materiality" of the chemical object (briefly commented in the introduction) and that deserves a future deep investigation.

The Different Reaction Mechanisms

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, a new nomenclature will be used here for the possible chemical reactions, with a simple illustration for each reaction mechanism. 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

- *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.

- *Open-bond reactions*: in the first molecule, a link containing i bonds opens itself to make j links of i/j bonds (here the first link "1-2" of the first molecule opens itself (i.e. frees one bond) and the first link of the second molecule breaks)

  1(2 2) + 2(3 3) → 1(2(3) 2(3) 2)

- *Close-bond reactions*: it is the reciprocal of the previous reaction - in the first molecule, the two first links of one bond merge to form one link of two bonds:

  1(2(3) 2(3) 2) → 1(2 2) + 2(3 3)

- *Changing-variance reactions*: in the first molecule, the first atom of variance i increase or decrease its variance (here decrease)

  1(4 4 4 4) → 1(4 4) + 4(4)

- *Re-organizational reactions:* it is just a re-organization of the links in the molecule:

  1(1(2 3) 3 3 3) → 1(3 3 3 3) + 1(2)

This re-organisation is often accompanied by a change in the variance of one of the atom (here the "1").

Simulating the Single-Link-Crossover and the Open-Bond Reactions

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 either the single-link-crossover or the open-bond reaction (random choice). In both cases, the link involved in each molecule (either

exchange or open) is the weakest link (each link receives a random energetic value).

3. Generate the new molecules in their normal form only if they do not already exist in the system.

A list of some firstly obtained molecules is given below:

1(3333), 1(2(2(3))333), 1(1(333)333), 1(4444), 1(3444), 2(2(4)3), 1(1(444)334), 1(1(333)1(333)1(333)1(333)), 1(2(1(333))333), 1(2(4)334), 1(2(1(444))444), 1(3344), 2(2(4)4), 1(1(333)1(444)1(444)1(444)), 1(1(333)1(333)1(333)2(2(1(333)))), 1(2(2(2(1(333))))333), 1(1(444)1(444)1(444)2(1(333))), 1(1(2(4)2(4)2(4))2(3)2(3)2(3)), 1(1(1(444)2(1(333))2(3))1(1(444)2(1(333))2(3))1(1(444)2(1(333))2(3))1(1(444)2(1(333))2(3))).

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

After several seconds of simulation you easily get molecules long of 47 atoms like the following one:

1(1(2(1(1(1(2(1(1(1(1(1(1(2(1(1(1(2(1(1(2(1(1(2(4)33)2))2)2)2))2)2)2)44)2)2)2))2)2)2))2)1(24)33)

or 93 atoms like:

1(1(1(444)1(444)2(1(1(444)1(444)1(444))))1(1(444)1(444)2(1(1(444)1(444)1(444))))1(1(444)1(444)2(1(1(444)1(444)1(444))))1(1(444)1(444)2(1(1(444)1(444)1(444)))))

You could substitute "1" with "C", "2" with "O" and "3" or "4" with "H" in order to obtain realistic organic molecules.

The Dynamics comes into play

Every chemical component, and thus molecule, is assigned an important attribute: its concentration (between brackets 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:

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

$$[E] = [F] \leftarrow k*[A]*[B]$$

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

$$[E] \leftarrow [E] + k*[A]*[B]$$

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

And in all cases:

$$[A] \leftarrow [A] - k*[A]*[B]$$
$$[B] \leftarrow [B] - k*[A]*[B]$$

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:

$$3(3) + 4(4) \rightarrow [2]3(4)$$ (the [2] is the stoichiometric coefficient)

Below (figure 7), 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.

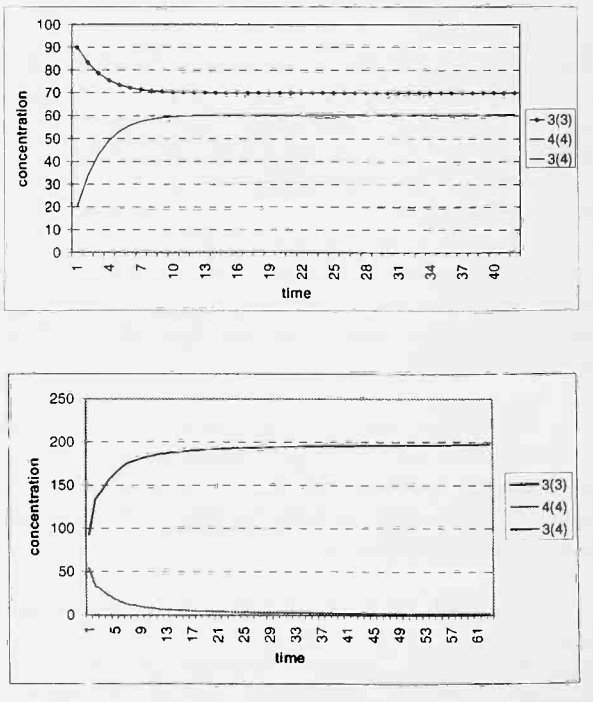


Figure 7: Time evolution of the concentration of the three interacting molecules: up – in the reversible case, down – in the irreversible case.

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 8 shows the evolution in time of 5 molecules among others.

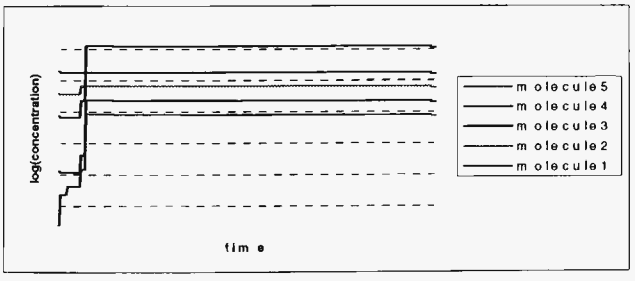


Figure 8: Time evolution of the concentration of the interacting molecules for the random interaction algorithm.

The flat line presence is due to the long period where nothing changes in the concentration of a specific molecule due to the birth of new molecules. The 5 molecules selecting in the plot are indeed the ones that re-enter in a reaction after a long period following their birth. 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 all molecules j (going from 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

The presence of the second loop going from "1 to i" in the loop maintains everything deterministic and just accounts for the symmetry of the reactions ("A reacts with B" is equivalent to "B reacts with A"). Although the best algorithmic version, because naturally leading to the analytical solution, the problem with such an algorithm is that everything rapidly exponentially explodes. 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 9.

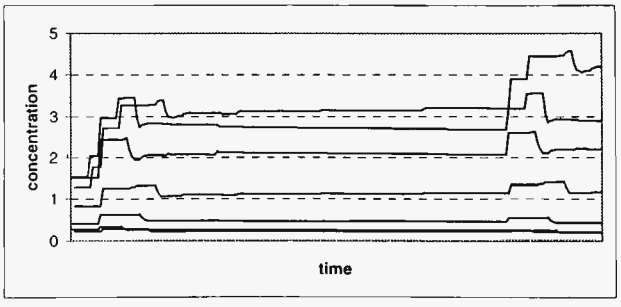


Figure 9: Time evolution of the concentration of seven molecules for the deterministic interaction algorithm

Interestingly enough, this version of the algorithm allows the detection of chaining and self-maintaining 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

self-maintaining reactions 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 (table 3). Indeed, the figure 7, 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).

In order to clarify further this point, let's suppose two trivial examples of chemical reaction. In the first example, the reactions are the following (let's suppose two simple first order reactions with reaction rates k_1 and k_2 and three initial conditions $[A] = [A]_0$ and $[B] = [C] = 0$)

$$A \rightarrow B \text{ and } B \rightarrow C$$

The plot in fig.10 shows how the molecule concentrations evolve in time in our simulation. There is no visible difference with the analytical solution given by:

$$[A] = [A]_0 \exp(-k_1 t)$$
$$[B] = k_1[A]_0(\exp(-k_1 t) - \exp(-k_2 t))/(k_2 - k_1)$$

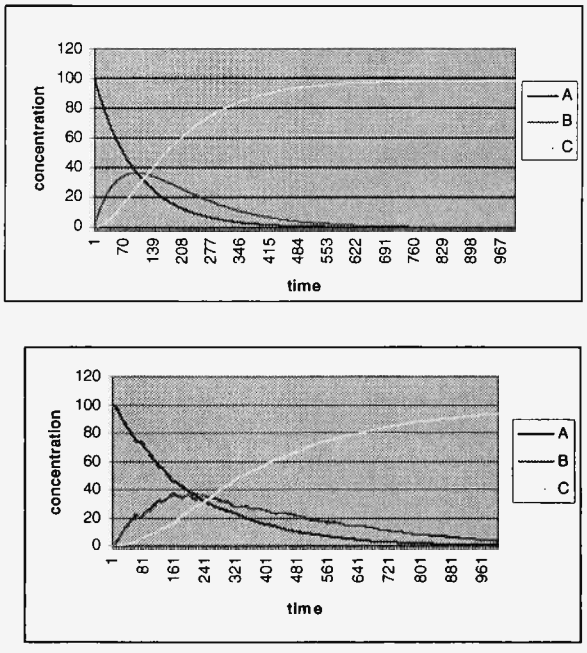


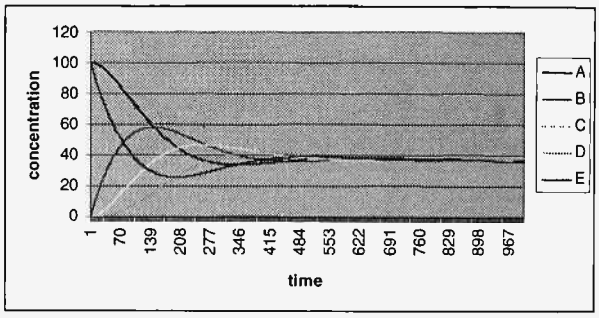
Figure 10: the concentration evolution in time of the molecules A,B,C - up): for the deterministic algorithm and down): for the random one

You can also see the plot obtained by a random choice between the two reactions and observe some noisy fluctuations that nevertheless still keep the global trends. Therefore for this simple reaction and, as a rule, as far as the deterministic algorithmic is computationally tractable, the analytical solution is obtained for sure. Randomness introduces noise in the plotting, and the amount of degradation is dependent on the complexity of the reaction scheme.

Let's suppose now a second more complex example consisting in a loop of 5 reactions (the last reaction closing the chain):

$$A \rightarrow B,\ B \rightarrow C,\ C \rightarrow D,\ D \rightarrow E,\ E \rightarrow A.$$

The "exact" plotting of the deterministic algorithm is given in fig.11. Adopting the random interaction algorithm described in table 2 gives the noisy plot shown in the same figure. Here the randomness induces a much severe degradation of the plotting.



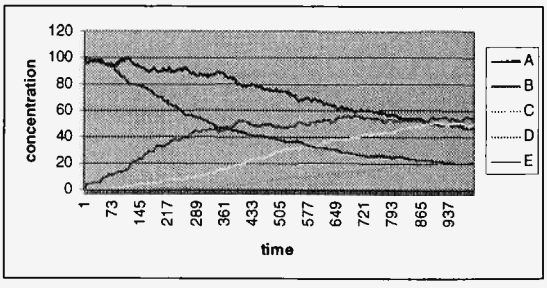


Figure 11: the concentration evolution in time of the molecules A,B,C,D,E - up): for the deterministic algorithm and down): for the random one

The intermediary "compromising" algorithm that could indeed appear as a form of time dilatation gives the most interesting plotting. The looping structure of the reaction induces the same type of artificial oscillation discussed previously. As a matter of fact this oscillatory pattern is the direct result of the loop and could indeed be exploited to detect this loop. In figure 12 you can see the evolution in time of the concentration of the molecule A for the three versions of the algorithm: determinist, "compromising" and the random. The oscillatory solution is the highest one since the concentration of A decreases slower than for the fully deterministic algorithm. During the first time steps of the

reaction, the two alternative simulations gives result very different form the analytical result.

Which version of the algorithm to adopt becomes a difficult question depending on the objective pursued by the simulations. A computer experiment aiming at the highest fidelity to the chemical reality should adopt the basic deterministic version. In such a case, it is capital to limit the number of possible reactions by a deeper analysis of which reaction turns out to be possible.

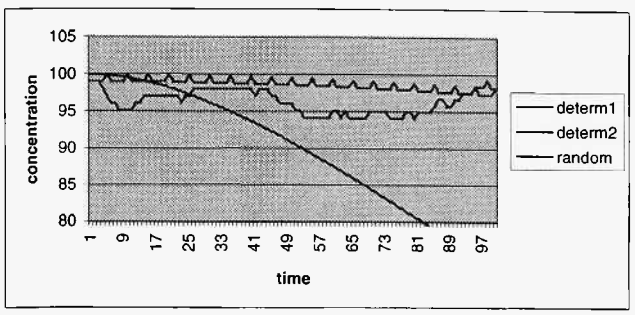


Figure 12: the concentration of the molecule A evolving in time for the three versions of the algorithm.

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. Easily detecting these self-maintaining structures could rather privilege the use of the second compromising version of the algorithm.

Finally the random version common in some artificial chemistry simulation already known in Alife aims at compensating for the absence in these works of the concentration variable. Introducing this variable make useless the random version all the more as it can considerably degrade the concentration evolution in time.

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.

References

1. Aho V. A., Ullman, J.D. (1995) : Foundations of Computer Science – Computer Science Press - W.H. Freeman and Company – New York.
2. 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
3. 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.
4. 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)
5. Dittrich, P. and Banzhaf, W. (1998): Self-evolution in a constructive binary string system. Artificial Life, 4(2):203-220.
6. Dittrich, P. (1999): Artificial Chemistries – Tutorial held at ECAL'99 – European Conference on Artificial Life, 17 September, 1999 – Lausanne, CH.

7. Farmer, J.D., Kauffman, S.A. and Packard, N.H. (1986) Autocatalytic reaction of polymers. Physica D, 22:50-67.
8. 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)
9. Fontana, W. and L.W. Buss (1996). The barrier of objects: From dynamical systems to bounded organization in Casti, J. and Karlqvist, A., editors, Boundaries and Barriers, pages 56-116. Addison-Wesley.
10. Holland, J.H. (1995): Hidden Order – How adaptation builds complexity – Helix Books – Addison Wesley Publishing Company (1995)
11. Kauffman, S. (1993): The Origins of Order: Self-Organization and Selection in Evolution – Oxford University Press
12. Eriksson, H-E, Penker, M. (1998): UML Toolkit – John Wiley and Sons