Chemlambda strings

Marius Buliga[*]

This version: 02.01.2018

Abstract

Chemlambda is an asynchronous graph rewrite automaton which uses a carefully selected family of graph rewrites of the kind encountered in Interaction Nets (IN). In this article is given a version of the graphs and rewrites which is more chemistry friendly. It is argued that real chemistry has enough place for accomodating chemlambda. The use of IN rewrite patterns in real chemistry, as templates of concrete chemical reactions, is an unexplored direction towards molecular computers. The simulations which validate chemlambda as a toy chemistry show that there is a big potential in this direction.

1 Introduction and background

Chemlambda strings is an alternative version of the chemlambda artificial chemistry [4] [5] [6] which is closer to real chemistry than the original. It is one more step ahead to the dream of building a real molecular computer in the sense described in this article [3] (brief idea introduced in [2]).

The aim is to build a material, chemical, asynchronous graph rewrite automaton, which works by uncontrolled, random graph rewrites (i.e. chemical reactions). A molecular computation, in this sense, consists of the transformation of an input molecule which, in a medium of either other "rewrite enzymes" or other simple molecules, enters in random, meaningless, uncontrolled reactions.

More specifically:

- we need a class of molecules made of specific elementary bricks, so that

[*]"Simion Stoilow" Institute of Mathematics of the Romanian Academy, PO BOX 1-764,014700 Bucharest, Romania, e-mail: Marius.Buliga@imar.ro , mbuliga@protonmail.ch

- there exists a dictionary which allows to translate an arbitrary (program+data) into a molecule from this class, so that

- the evolution of this molecule, under random chemical reactions, either with "rewrite enzymes" or other ingredients, is equivalent with the execution of the (program+data), so that

- despite randomness and lack of control over the order of reactions, the molecule attains eventually a state (i.e. transform predictively into another molecule) which represents via the same dictionary the output of the computation of (program+data).

If possible, the applications of such a chemical graph rewrite automaton would be huge. Indeed, literary "Turing universal" would translate into "capable of anything (which can be done via an algorithm) under no external supervision".

This is such a huge goal that even proof of principle proposals are valuable. I believe chemlambda is one of them. In this article "chemlambda" denotes the formalism (graphs, rewrites) from [3], and the algorithm is the one embodied into the chemlambda-gui repository [6].

For the interested reader, or even better for the interested researcher willing to try an Open Science validation approach, the repository [6] is the main reference to chemlambda. The repository contains a collection of more than 400 chemlambda molecules. The collection of simulations [7] contains simulations of most of these molecules, many of those used and briefly commented in the chemlambda collection hosted on G+ [8].

This article does not concentrate on the mathematical and computing background, on purpose, in order to let the main idea uncluttered. However, there are several remarks to be made.

The graphs from chemlambda (called "molecules") are, mathematically, oriented ribbon graphs. The graph rewrites of chemlambda have the general form encountered in Interaction Nets [12]. Apart from this, in chemlambda there is no constraint on the graphs-molecules to represent a particular class of "correct" graphs, for example graphs which could represent lambda terms. The algorithm of graph rewrites applications, i.e. the reduction algorithm, is the most simple, turning chemlambda into an asynchronous graph automaton. Compared with the main idea that this particular concrete automaton can be done with real chemistry, there is not much appeal to use verified but too abstract mathematical tools and relations, which bring unnecessary and sometimes misleading backgrounds to a concrete problem.

In chemlambda, the graphs used model molecules, not reaction networks. These graphs are not DAGs, nor there is a preferred embeding into space, nor there is a time direction which defines a flow of something along the edges of the graph. The nodes of the graphs are not gates, or functions, they are elementary molecular bricks.

There exist classical models of computation which use a chemical analogy, first to come to mind are the ALgorithmic CHEMistrY of Fontana and Buss [10] and the Chemical Abstract Machine of Berry and Boudol [1]. There are however different than chemlambda. In Alchemy there is used an analogy with lambda calculus, which is a model of computation with terms and operations between them, called aplication and abstraction. In Alchemy the application is a chemical reaction ($A + B \rightarrow AB$) and abstraction is a reaction site, therefore application and abstraction are different things, conceptually. Not so in chemlambda, where, by the dictionary alluded previously, application and abstraction translate into nodes. In the CHAM are used multisets of molecules, called "solutions", enclosed by membranes, which can be heated up or cooled down. In chemlambda we are interested in individual molecules, which evolve by random chemical reactions and still they "achieve their goal", in the sense that despite the randomness, eventually a predictable configuration is attained.

The other viewpoint, one clearly grounded in real chemistry, is also avoided here. I am not a chemist. However, the main idea is that there should exist real molecules and a very small class of templates for chemical reactions which can do what chemlambda does on paper (in a computer, actually). The simulations largely support this idea, in the sense that this particular model, or perhaps a variant of it, once realized, will prove it is possible to model molecules which suffer long strings of random chemical reactions, despite the fact that standard treatments lead us to exponential explosions of possibilities. Not any molecule and not any kind of reactions would be amenable to this treatment. But, as suggested in [2], what a wonderful experiment would be the one which, using chemlambda, one can compute, with real chemistry, a value of the Ackermann function. For example, Ackermann(2,2)=7 is computed by a sequence of 32 random reactions.

2 Sticks and rings

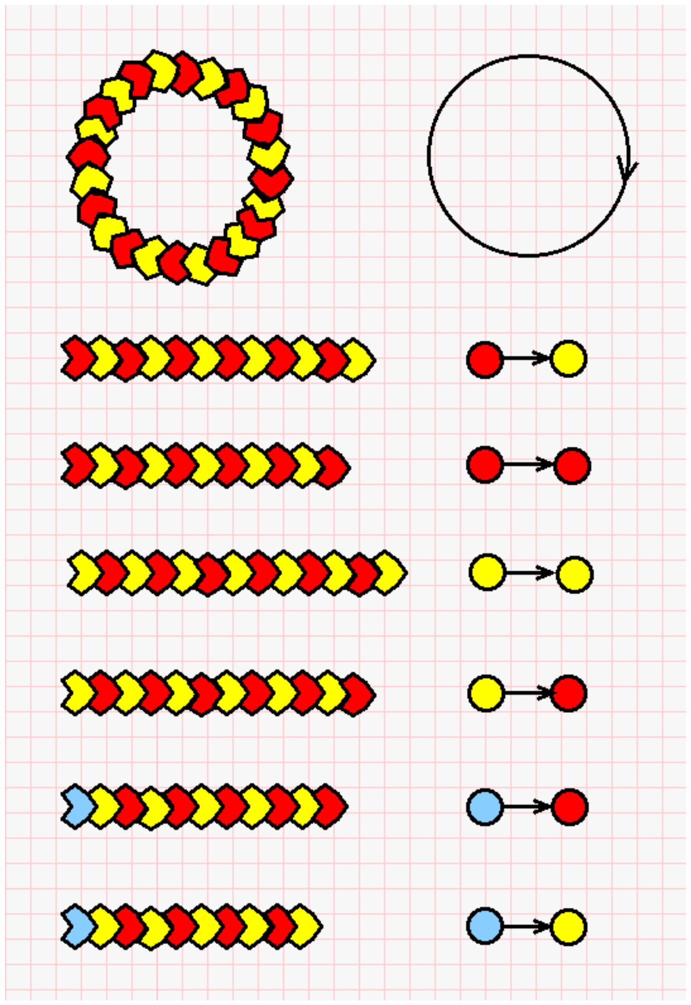
Suppose we have a provision of asymmetric molecules, of three types, described as red, yellow and blue. They are asymmetric in the sense that we may identify a part which is called, conventionally, "in" and a part which is called "out", so that there is an unambiguous way to tell the "in" from the "out", as well as the possibility to identify a sense from "in" to "out". The color (red, yellow, blue) may be imagined as representing different chemical composition, or perhaps a part of the molecule which is different, in three ways.

Among the molecules which satisfy this description are RNA bases [13], Figure 1. Geometrically, any such base is modeled as a triangle, with edges which represent different types of bonds. In graph theoretic terms, any such base is a trivalent node, with a given numbering (or naming) of the (half-)edges incident to the node. There is a Watson-Crick edge, a Hoogsteen edge and a sugar edge. A pairing of two RNA bases is described by a pair of half-edges, like for example Watson-Crick/Watson-Crick. There is even more geometric freedom in a pairing, which can be also a *Cis* or a *Trans* pairing.

This example shows that there is more than enough diversity of geometrical structure available from chemistry. However, of course that eventually a real chemical realization of the chemlambda formalism can appear only after serious input from chemistry researchers.

With this understanding, let's pursue with this probably very simplified chemistry analogies. These molecules can connect one with another in the following ways. As concern the red and yellow molecules, they can make long chains, or strings of alternating red and yellow molecules, made by connecting the "out" of a molecule with the "in" of the next one, in the sense defined by their asymmetry. A blue molecule can connect via the "out" part with the "in" part of a red or yellow molecule. Any other connection between "in" and "out" parts is prohibited.

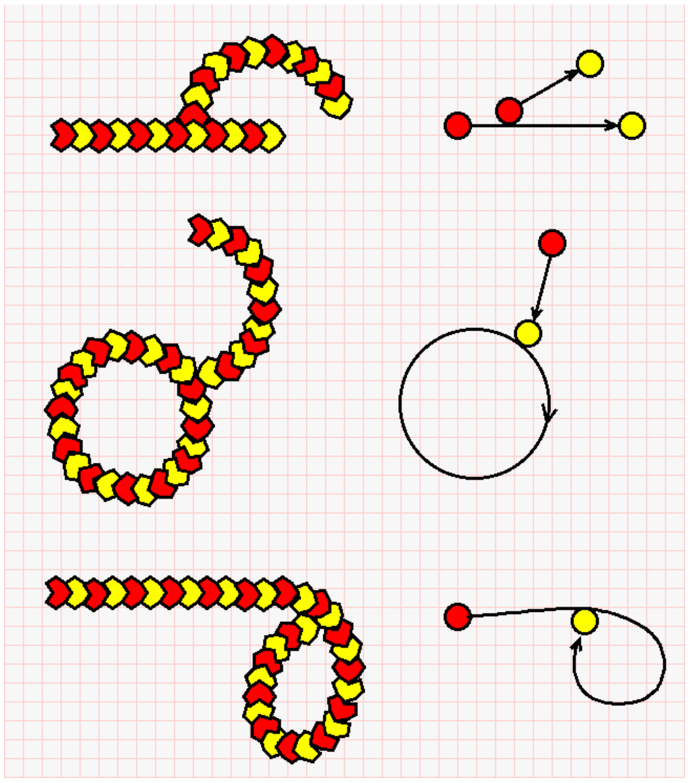
These strings of molecules can be described by indicating the color of the ends of the strings, if any. The possible strings are described in the next picture .



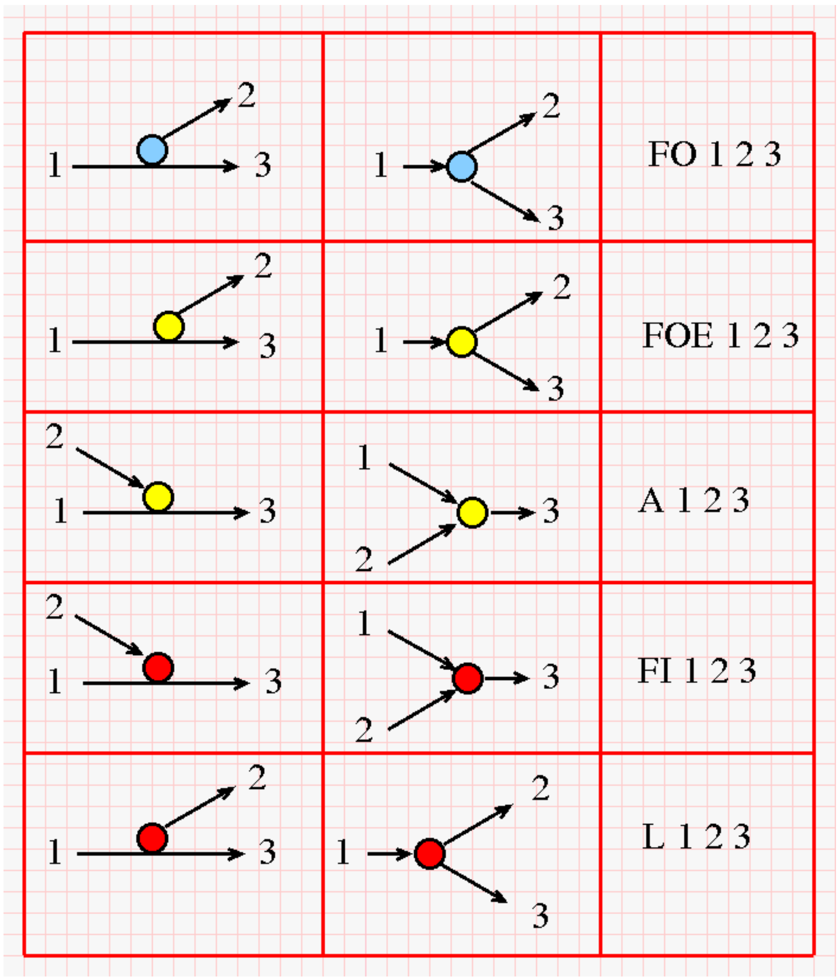
The red and yellow molecules have also a different part, let's call it "middle". Any two red or yellow molecules can connect via a "middle" - "in" or a "middle" - "out" connection, provided that the molecule which brings the "middle" part is not an end of a string. Any blue molecule can connect it's "in" part with a "middle" part of a red or yellow molecule which is not an end of a string.

Another variant is that we can simply suppose that we have this strings, either open (i.e. "sticks") with two sticky, colored ends, or closed (i.e. "rings"). Either way, the ingredients of this simplified chemistry will be these sticks and rings. The sticks ends can adhere to other sticks or rings.

This gives more complex structures, made of one or several strings (sticks or rings). We describe these connections as a tangency between an extremity of one string and (the bulk of) another string.



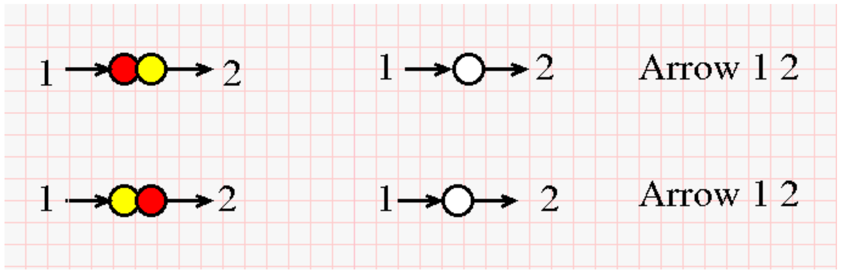
Each "middle"- * connection encodes a trivalent node of an oriented trivalent ribbon graph. More precisely, there is the following correspondence between the possible "middle"-* connections of strings and the nodes which compose chemlambda graphs [3].



The figure has 3 columns. In the first column is a possible connection of strings, with the strings pieces around the connection numbered with "1", "2", "3". In the second column we see the corresponding chemlambda node. In the third column is the "mol" encoding of the node as used in the mol notation from chemlambda. With respect to the color codes used in chemlambda, here we have a slightly different convention, namely: (first line) a fanout "FO" node is colored with blue, instead of green in the original version, an application "A" node is colored with yellow, instead of green in the original version.

As for the auxiliary "Arrow" 2-valent nodes which are used in the original chemlambda, they are trivial here, because of the way a string is built.

Therefore there will be no "Arrows" in this version, nor the corresponding COMB rewrites of chemlambda. As explained in the next figure, an "Arrow" node just expresses the fact that a string can be seen as two strings connected end-to-end.



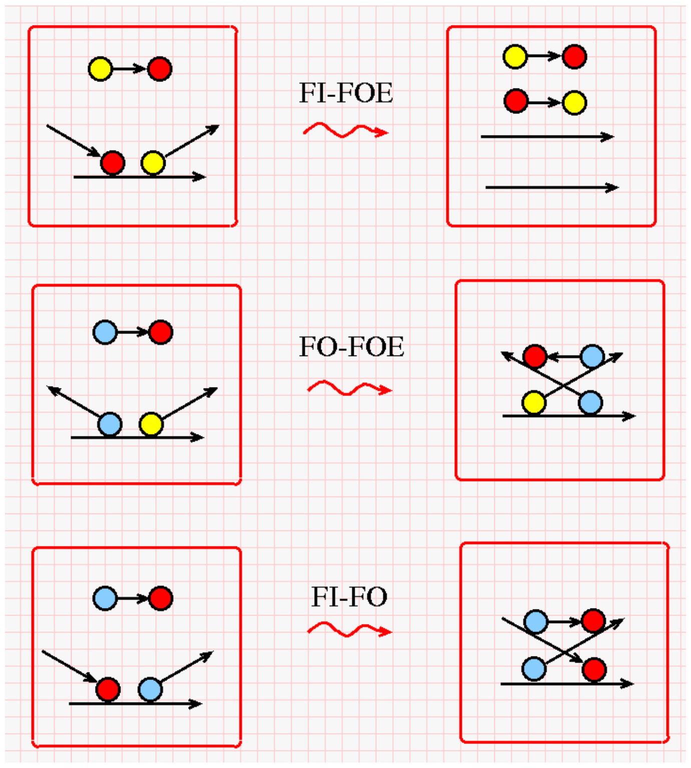
3 Reactions

The graph rewrites of the original chemlambda are seen as chemical reactions of the form:

(Pattern before the graph rewrite) + (rewrite enzyme) $\rightarrow$ (Pattern after the graph rewrite) + (rewrite enzyme)

For the interested programmer, see the implementation of these reactions in the "chemlambda-hask" Haskell API of J. King, in Enzyme.hs from [11] this branch of his github repository.

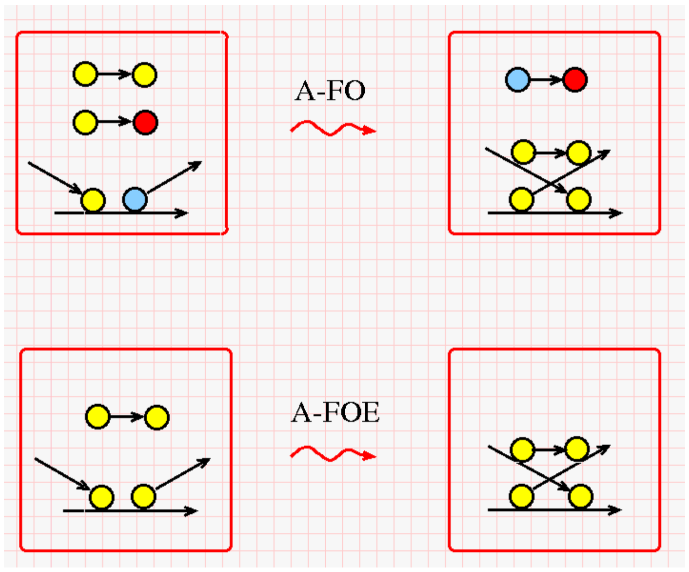
An interesting feature of the chemlambda strings version described here is that we can renounce at these invisible rewrite enzymes. Moreover we can make the chemical reactions implementing the chemlambda rewrites to conserve the strings, equivalently to conserve the nodes and arrows. This is very attractive from the point of view of real chemistry.



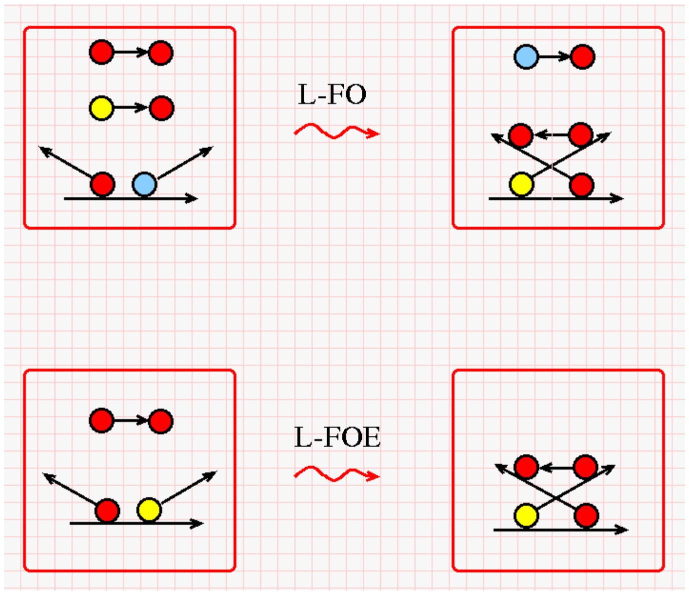
Some comments are needed. The FI-FOE rewrite in the original chemlambda uses "Arrow" elements, which are then erased by the COMB rewrites. Here we see that a free yellow-red string acts indeed as an enzyme (because it is present in both sides of the reaction). Instead of "Arrows", the reaction result contains also a free red-yellow string, which will appear in other chemical reactions.

The FO-FOE and FI-FO rewrites are in the family of "distributive", or DIST rewrites of chemlambda. Here they appear as a free blue-red string which (1) permutes two neighboring string ends which sit on a string, then (2) the free blue-red string glues itself on the permuted strings.

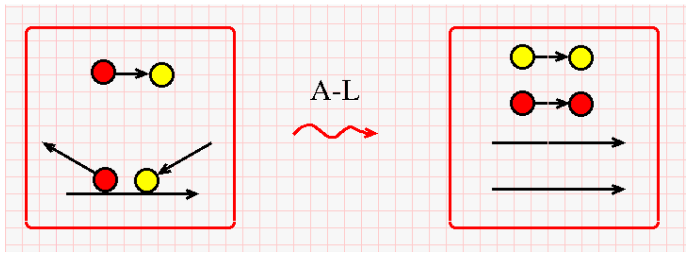
The DIST rewrites for the application "A" node in chemlambda are given in the next figures. The A-FO rewrite needs a free yellow-yellow and a free yellow-red string and it produces a free blue-red string. Otherwise is the same permute-then-glue reaction. The A-FOE reaction needs only a free yellow-yellow string.



The DIST rewrites for the lambda "L" node in chemlambda have a similar pattern. This time red-red and yellow-red free strings are needed and a blue-red free string is produced.



Finally, here is the A-L rewrite (inspired from the famous BETA rewrite from lambda calculus).



It is almost similar with the FI-FOE rewrite, but the A-L reaction needs a free yellow-red string and produces a free yellow-yellow string and another free red-red string. No "Arrow" nodes are needed.

4 Further notes

The mathematical treatment of this model is left for an accompanying paper and program repository. Just to have an idea, the proper treatment is in terms of permutation automata. Stochastic ingredients, as well as other kinds of reactions, will be considered and explained in the accompanying material.

References

[1] Berry, Gérard, Boudol, Gérard, The chemical abstract machine (1992), Theoretical Computer Science **96**, 1, p. 217 - 248

[2] (github) Buliga, Marius , Molecular computers (2015)

[3] (journal) Buliga, Marius (2015). Build a molecular computer. http://doi.org/10.5281/zenodo.16018

[4] (figshare) (arxiv) - Buliga, Marius (2013): Chemical concrete machine. figshare. https://doi.org/10.6084/m9.figshare.830457.v1

[5] (MIT Press free download) (arxiv) Buliga, M. , Kauffman, L.H. , Chemlambda, universality and self-multiplication (2014), Artificial Life 14, Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems Edited by Hiroki Sayama, John Rieffel, Sebastian Risi, Rene Doursat and Hod Lipson, p. 490 - 497

[6] (github) Buliga, Marius , The Chemlambda repository. For using the repository see the README of the active branch.

[7] (figshare) Buliga, Marius (2017): The Chemlambda collection of simulations. figshare. https://doi.org/10.6084/m9.figshare.4747390.v1

[8] (G+) Buliga, Marius (2016-2107), The chemlambda collection on G+

[9] (journal) (arxiv) Buliga, Marius, Graphic lambda calculus. Complex Systems 22, 4 (2013), 311-360

[10] Fontana, Walter, Buss Leo W., The Barrier of Objects: From Dynamical Systems to Bounded Organizations, in: Boundaries and Barriers, J.Casti and A.Karlqvist (eds.), pp.56116, Addison-Wesley, 1996

[11] (github) King, J. , The Chemlambda-hask repository

[12] Lafont, Yves, Interaction nets (1990). Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM: 95108

[13] Leontis, N.B., Westhof, E. , Geometric nomenclature and classification of RNA base pairs, RNA (2001), 7:499-512 online version