Graph rewrites, from graphic lambda calculus, to chemlambda, to directed interaction combinators

Marius Buliga

Institute of Mathematics, Romanian Academy
P.O. BOX 1-764, RO 014700
Bucureşti, Romania

Marius.Buliga@imar.ro , mbuliga@protonmail.ch

06.07.2020

Abstract

Here I report about the modifications of and relations between graphic lambda calculus, various formalisms which appeared under the name chemlambda and a version of directed interaction combinators.

This is part of the study and experiments with the artificial chemistry chemlambda and the relations with lambda calculus or interaction combinators, as described in arXiv:2003.14332 and available from the entry page chemlambda.github.io [12].

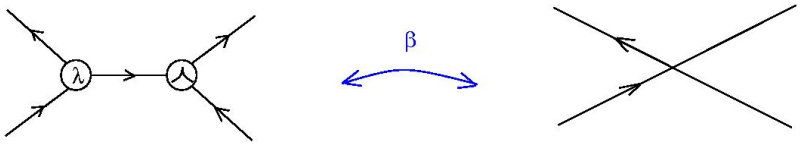
Context

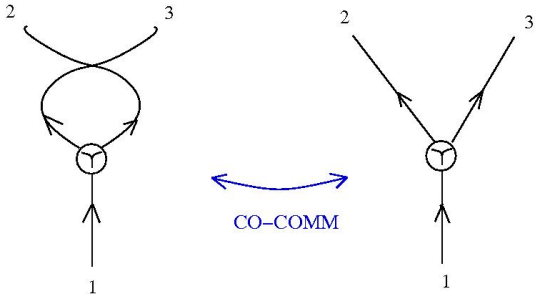
The chemlambda project context and relevant previous work are explained in Section 4 (About this project) of [4] arXiv:2003.14332 See also, for more mathematical background, the presentation [6] Emergent rewrites in knot theory and logic.

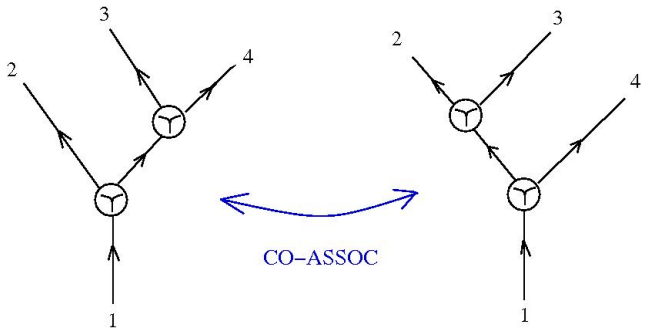
Here I report about the modifications of the various formalisms which are related to chemlambda. The online varsion of this article, which will be kept up to date, is Graph rewrites, from emergent algebras to chemlambda. See also the site of all chemlambda projects.

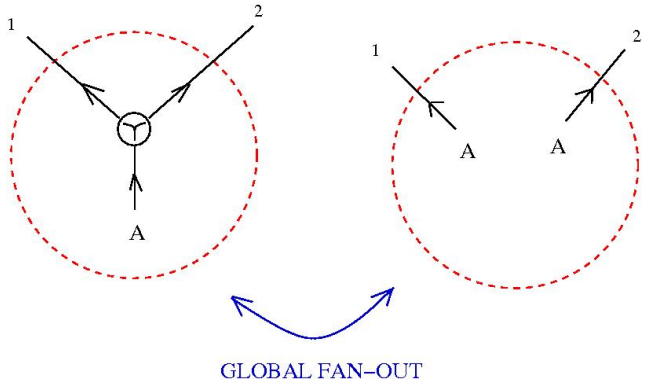
1 Graphic lambda calculus

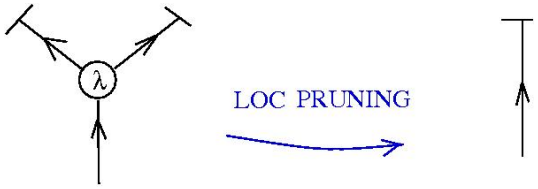
Introduced in [3] arXiv:1305.5786, graphic lambda calculus has the following doupbel push-out (DPO [15]) graph rewrites:

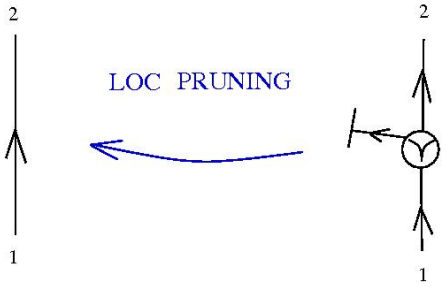

λ
β

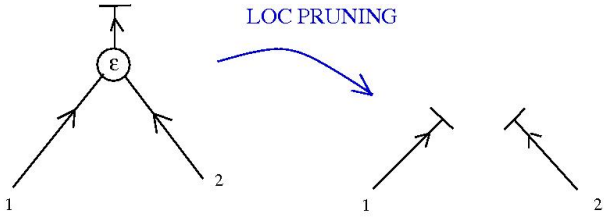

2
3
1
CO–COMM

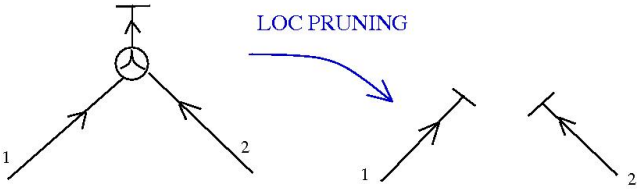

3
2
4
1
CO–ASSOC

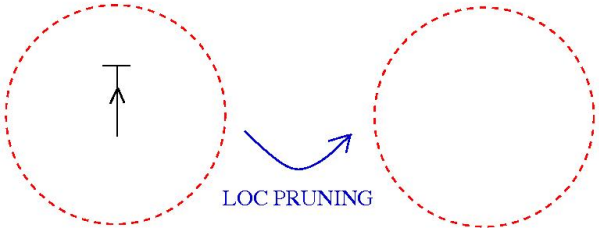

1
2
A
A
GLOBAL FAN–OUT

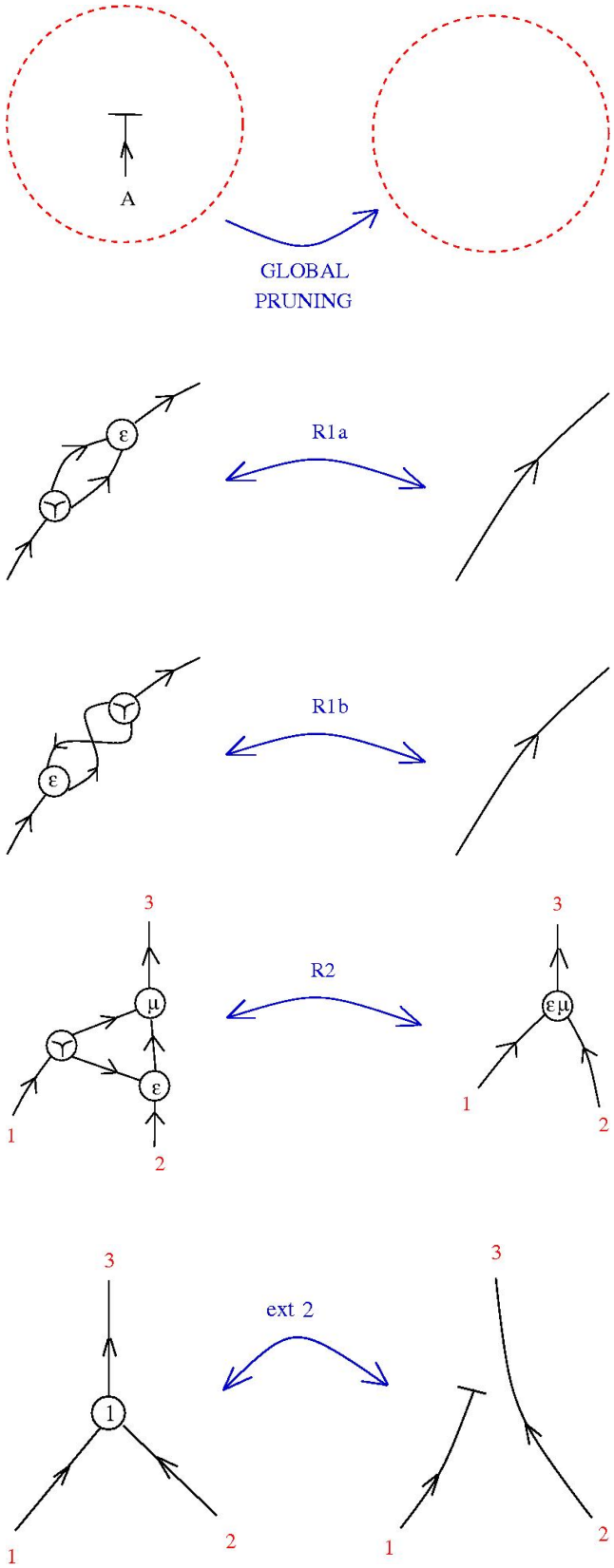

λ
LOC PRUNING


2
1
LOC PRUNING
2
1


ε
1
2
LOC PRUNING
1
2


1
2
LOC PRUNING
1
2

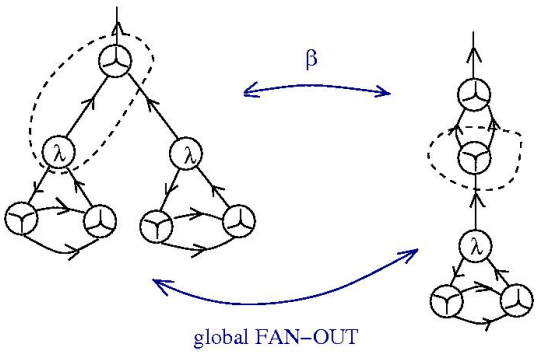

LOC PRUNING



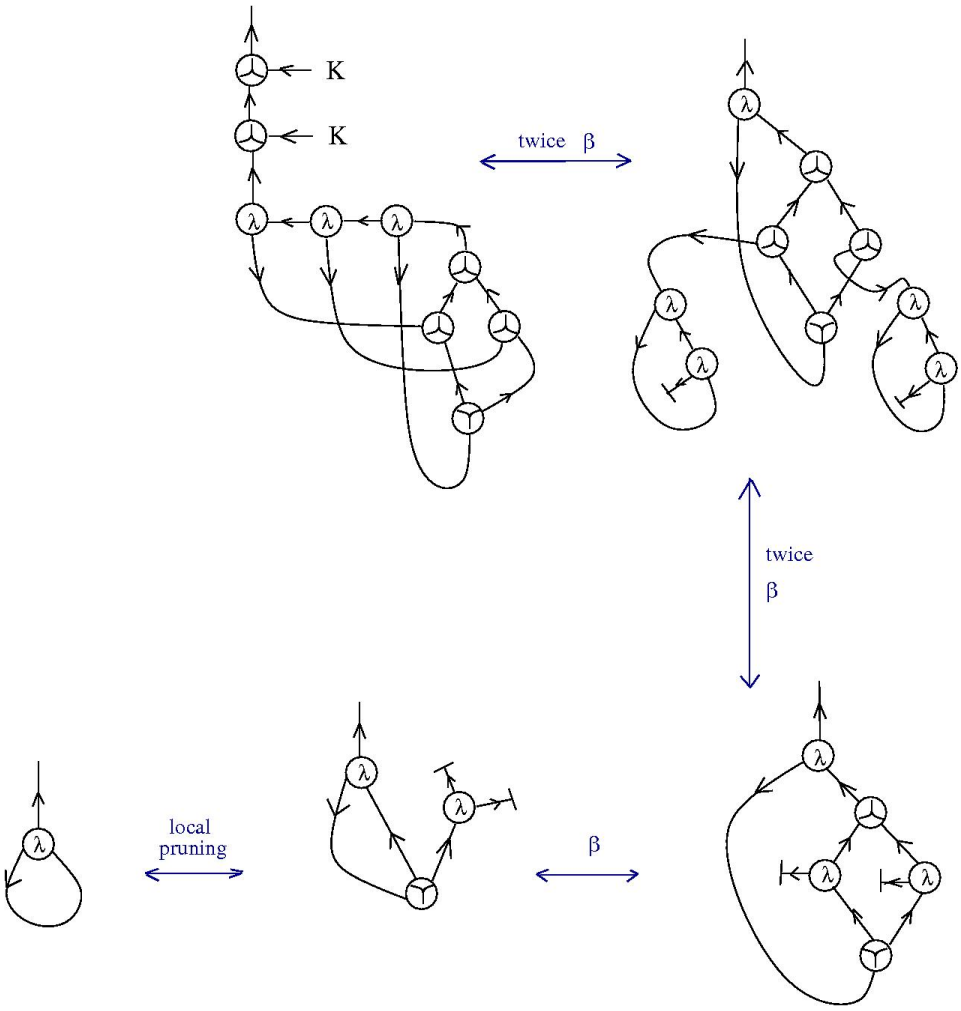
Graphic lambda calculus does not have an algorithm of application of rewrites.

There is an algorithm for conversion of untyped lambda terms into glc graphs. Here are two examples of reductions:

- reduction of the Omega combinator

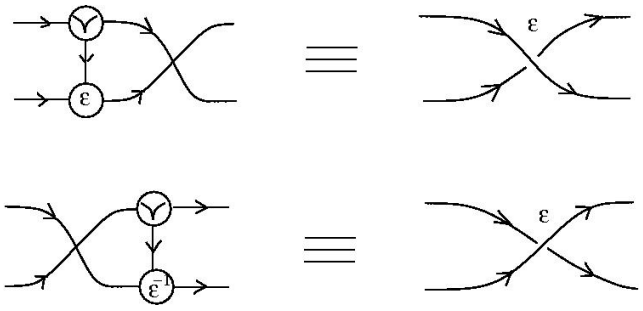


- reduction of the term SKK

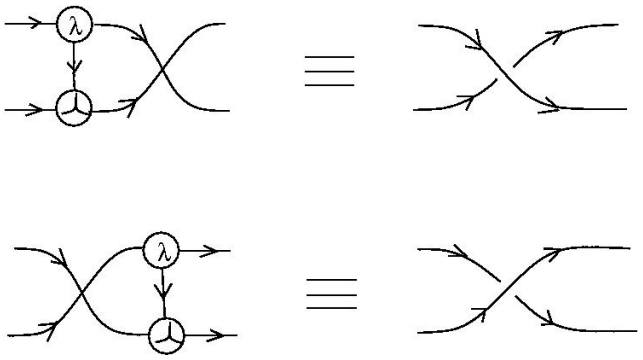


Graphic lambda calculus is interesting because it can also represent the graphical version of emergent algebras via decorated tangle diagrams explained in arXiv:1103.6007. It can therefore be used to represent and compute (reduce) differential calculus in metric spaces endowed with dilation structures arXiv:0810.5042.

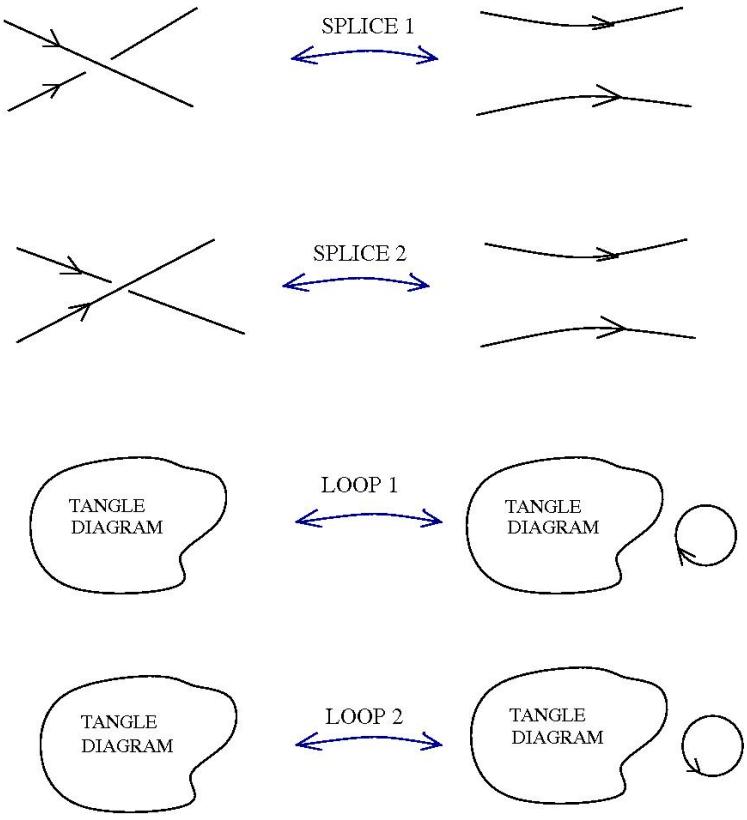
This is done via the identification of decorated crossings as



Another, different identification of undecorated crossings, which can be done with GLC, uses the application and abstraction nodes



Under this identifications the equivalent of the beta rewrite in terms of tangle diagrams becomes equivalent with the "splice" and "loop" graph rewrites.



Both these identifications apply for a notion of tangle diagrams which differs from the usual one. Namely we use tangle diagrams which are not constrained to be planar graphs. More precisely, here a tangle diagram is an oriented port graph in the sense of Bawden [1] [2], with nodes which are 4-valent and oriented links. The nodes can be either undecorated, i.e. of the two kinds of oriented crossings, or decorated, i.e. the two kinds of decorated crossings are supplementary decorated with an element of a commutative group.

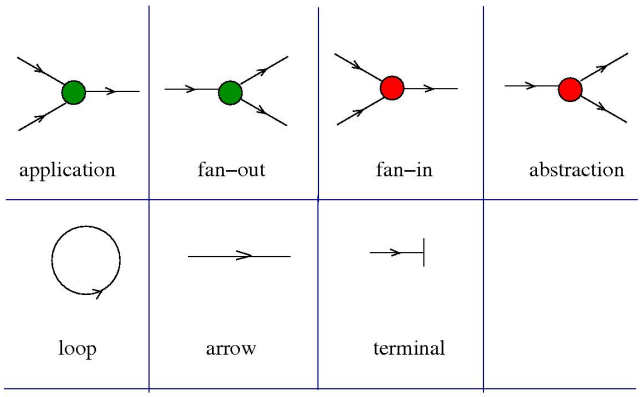
The difference from usual (oriented and with decorated crossings) tangle diagrams is that as graphs these diagrams are not supposed to be planar. By Kuratowski theorem, a graph is planar if it does not contain certain local patterns. In the terms of local machines explained in [5] Alife properties of directed interaction combinators vs. chemlambda we can't detect with a local machine the absence of a local pattern in a graph, only the presence of it.

In joint work with L.H. Kauffman [7] arXiv:1312.4333, in section 5 Kauffman calls for a topological version of GLC, based exclusively on manipulations of tangle diagrams.

2 Chemlambda v1 (chemical concrete machine)

Introduced in [9] arXiv:1309.6914, chemlambda v1 or the chemical concrete machine is a modification of GLC which contains only local rewrites. It also proposes for the first time to see the graphs as molecules and the rewrites as chemical reactions mediated by rewriting enzymes. Beyond that, chemlambda v1 still lacks a clearly defined algorithm of rewriting.

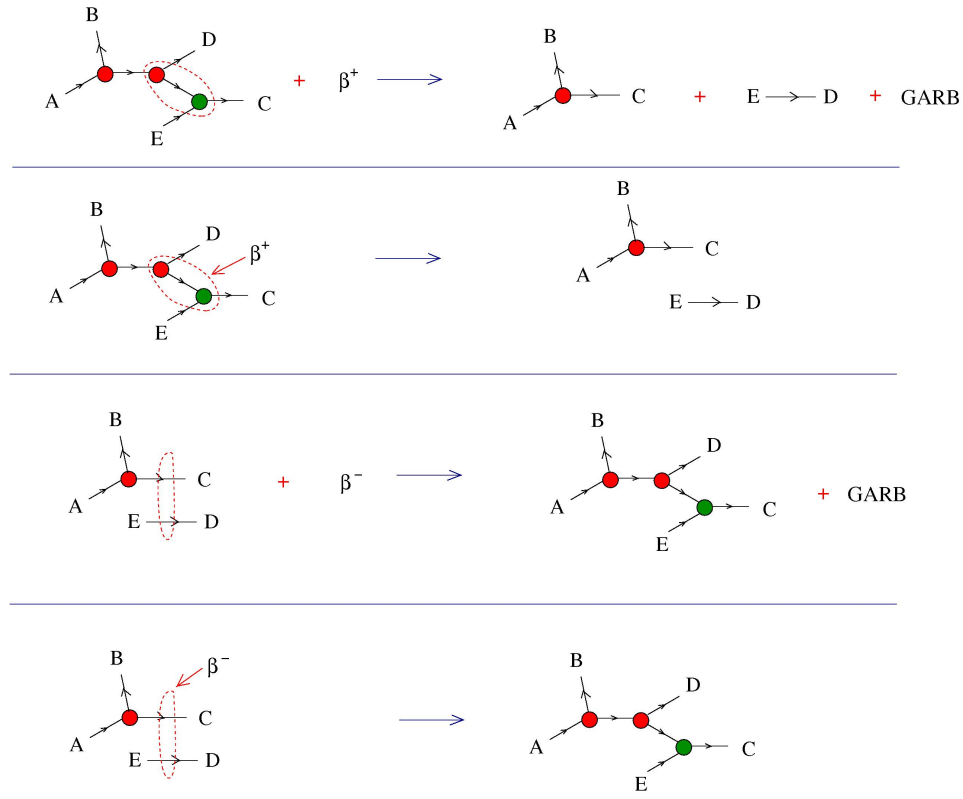
The elements of chemlambda v1 are



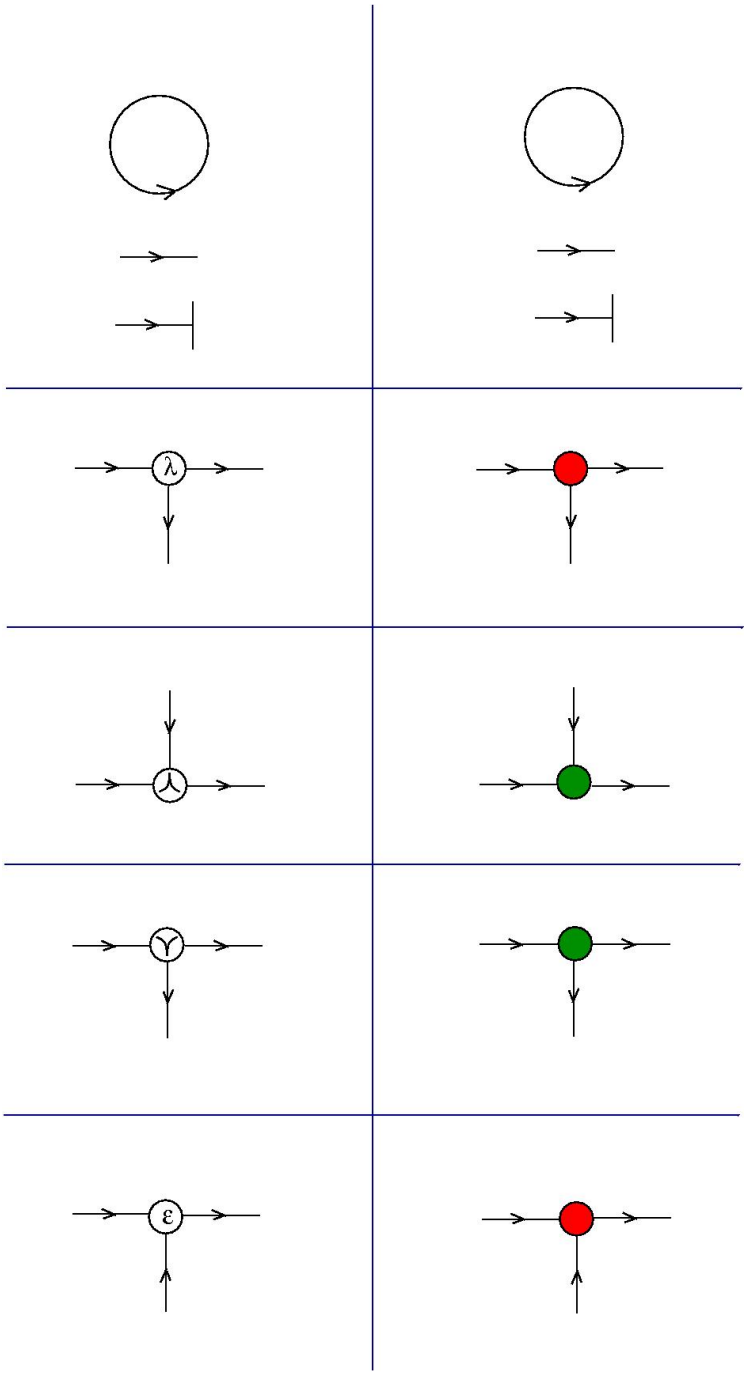
A B C ... GARB

other molecules		garbage (inactive molecules)
β^+	β^-	beta enzymes
ϕ^+	ϕ^-	FAN–IN enzymes
χ^+	χ^-	CO–COMM enzymes
α^+	α^-	CO–ASSOC enzymes
δ^+	δ^-	DIST enzymes
π^+	π^-	PRUNING enzymes

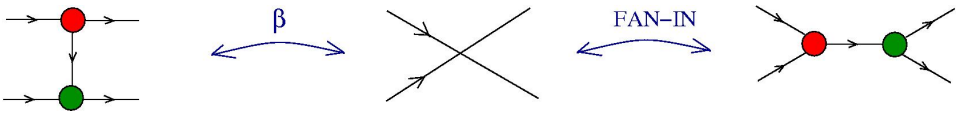
The graphs are called molecules and the rewrites are mediated by enzymes.

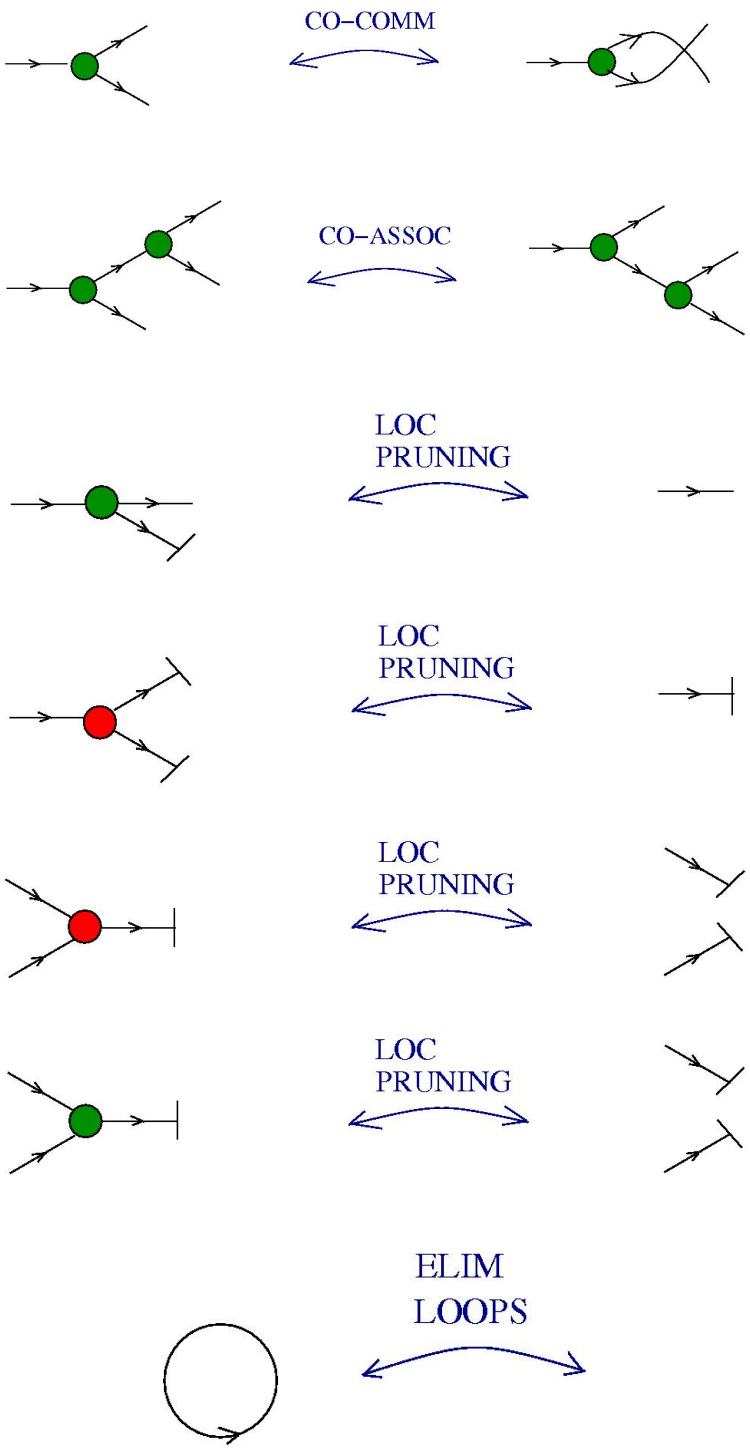


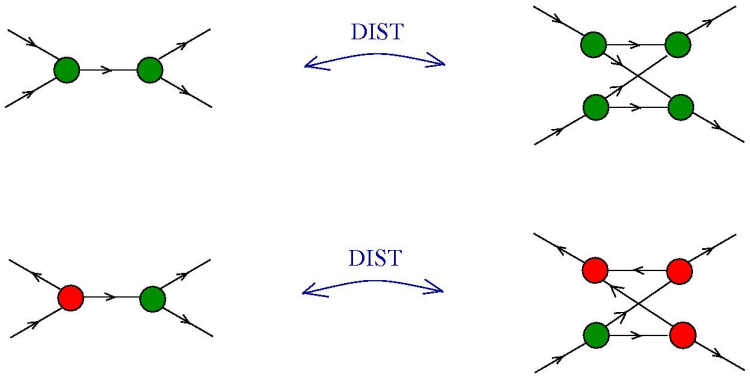
The elements of GLC as translated into chemlambda are:



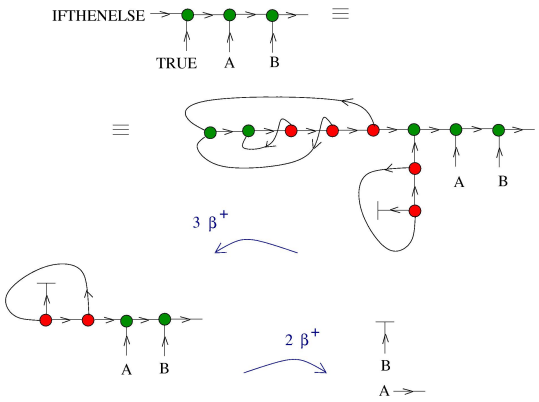
The rewrites are:




CO−COMM
CO−ASSOC
LOC
PRUNING
LOC
PRUNING
LOC
PRUNING
LOC
PRUNING
ELIM
LOOPS

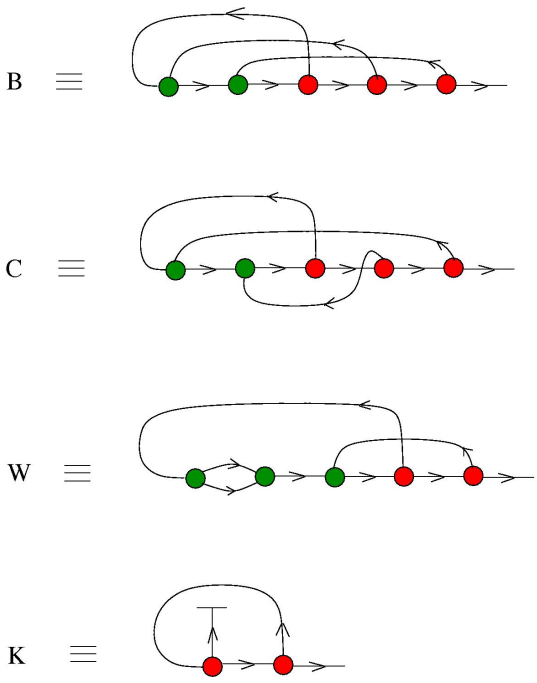


With these rewrites we can do some lambda calculus reductions, by using the same parsing of lambda calculus terms to (chemlambda) graphs as in GLC. For example the ENELSE lambda term.

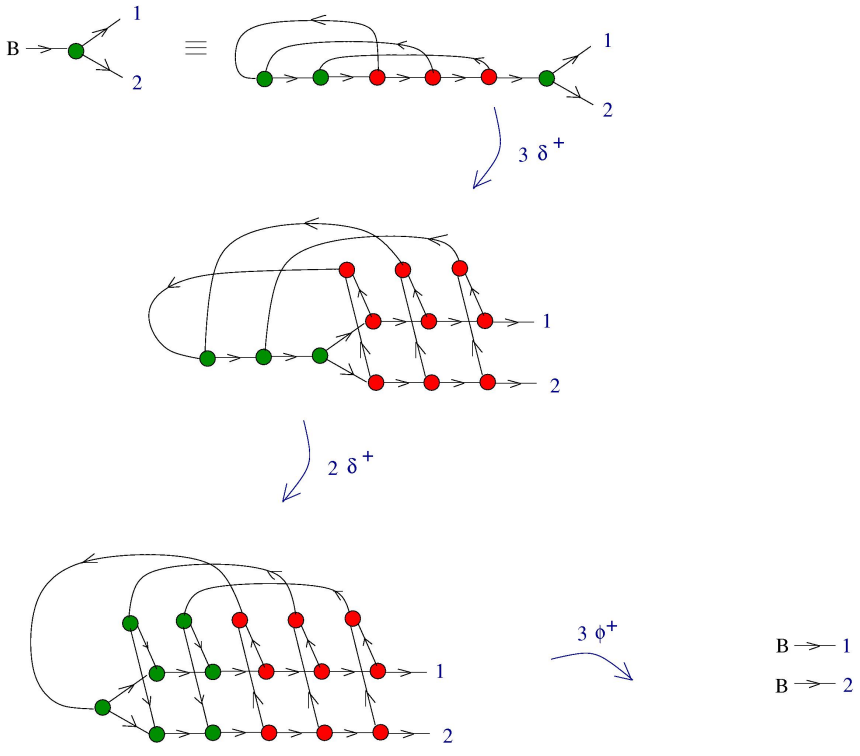


A rewrite which is introduced later in the article is DISENTANGLE (as a pattern it appears later as SHUFFLE)

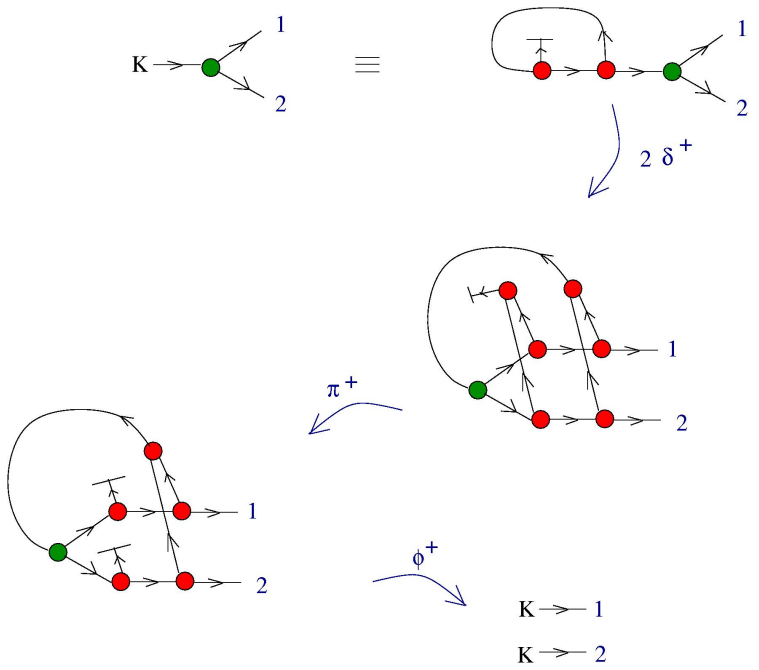
With this supplementary rewrite I can prove that the BCKW combinators system



can be implemented, in the sense that the combinators can duplicate (example here for B)



and that all needed reductions among B,C,K,W combinators can be done (example here for K)

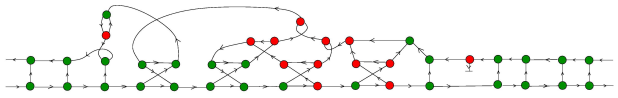


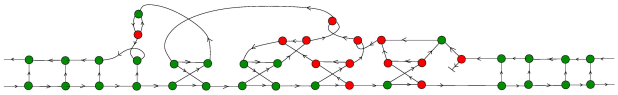
This graph rewrite system has conflicting rewrites, but it has no algorithm of application of rewrites. Especially the status of DISENTANGLE is unclear. This will be solved in chemlambda v2.

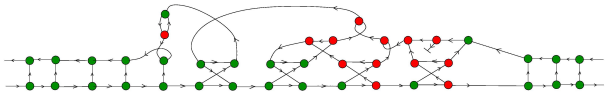
The first published (in the usual ways) article which contains the name "chemlambda" is joint with L.H. Kauffman, in the ALIFE-14 conference, arXiv:1403.8046. Here Kauffman pushes again for a more topological version Kauffman' slides [11] which mixes tangle diagrams with chemlambda nodes, in order to be able to do computations (in the logical sense).The emergent algebras part of GLC, which is still present in chemlambda, used such tangle diagrams manipulations for differential calculus computations, instead. In these articles the emergent algebras aspects and interest are hidden.

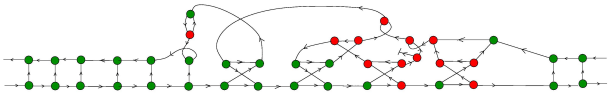
Together with Kauffman, we experimented a lot with the power of chemlambda v1. Our essays on paper were not going too far, due to the lack of a program which could make much easier such exploration. In the background this was because of a lack of a clear rewrite application algorithm.

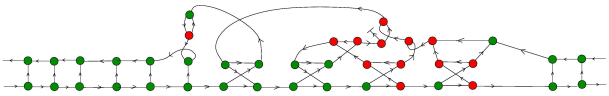
During this period I became aware of the existence of chemlambda quines. See as an example the tedious manipulation of reductions for the predecessor of a Church number taken from this chorasimilarity post

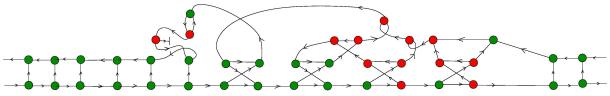


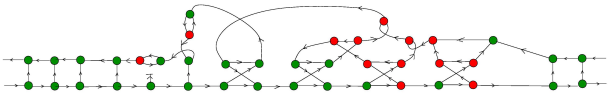


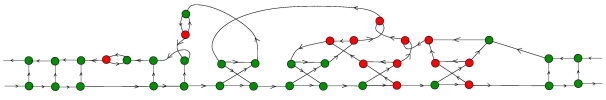












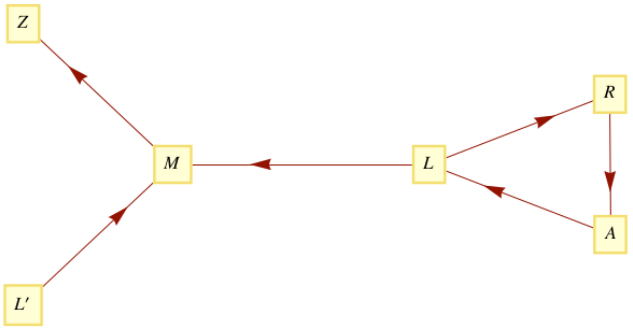
Meanwhile Kauffmann started to use Mathematica for doing chemlambda v1 reductions,

as described in his slides [11].

```
In[109]:=
t = H[L → R] H[R → A] H[A → L] H[L → M] H[L' → M] H[M → Z];
Graf[t]
PGraf[t]
Graf5[t]

Out[110]= H[R → A] (H[L' → R] H[A → Z])
```

Out[111]=



Out[112]=

He introduced the "Arrow" 2-valent node, which is useful for easy application of several rewrites in the same time.

3 Chemlambda v2

is an artificial chemistry, in the sense that it is a purely local graph rewrite system together with an algorithm of applications which can be done by local machines. This notion of artificial chemistry is proposed in [17]. See arXiv:2005.06060 for more explanations.

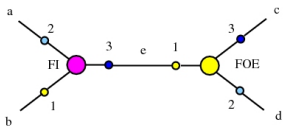
Motivated by the first programming essays by Kauffman for chemlambda v1, I wrote the needed programs for chemlambda v2, in awk and html+d3.js used for visualization. The first chemlambda project site [13] contains demonstrations made with these programs from the github repository chemlambda-gui [14].

The latest chemlambda project site, which contains several repositories from various contributors, is this [12]. The project received help from several contributors, or validators, therefore now we can reduce chemlambda molecules in haskell, python or javascript, as you can see in the more recent demonstrations.

Chemlambda can now be compared with Interaction Combinators [5] and there is a lambda calculus to chemlambda parser and reducer [8].

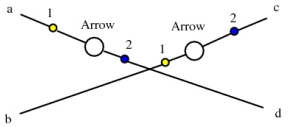
The chemlambda v2 rewrites solve the problem of the curious DISENTANGLE rewrite by introducing two fanout nodes (FO and FOE) instead of one and by a modified set of graph rewrites. These are the following. We use the mol notation for clarity and we describe the rewrites by giving the LHS and RHS patterns:

LHS



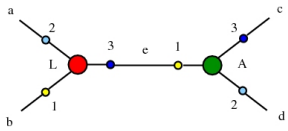
FI b a e
FOE e d c

RHS



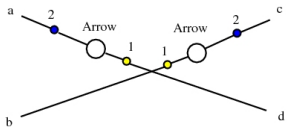
Arrow a d
Arrow b c

LHS



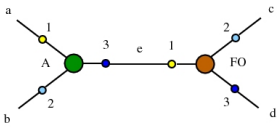
L b a e
A e d c

RHS



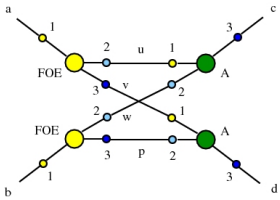
Arrow d a
Arrow b c

LHS



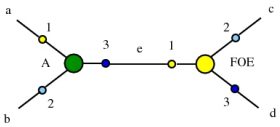
A a b e
FO e c d

RHS



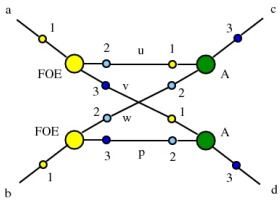
FOE a u v
FOE b w p
A u w c
A v p d

LHS



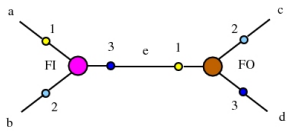
A a b e
FOE e c d

RHS



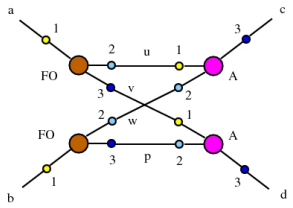
FOE a u v
FOE b w p
A u w c
A v p d

LHS



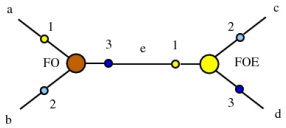
FI a b e
FO e c d

RHS



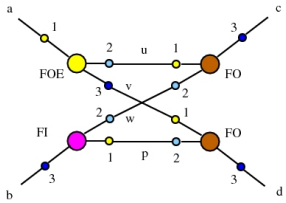
FO a u v
FO b w p
FI u w c
FI v p d

LHS



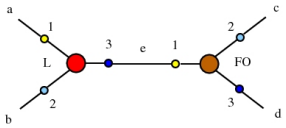
FO a b e
FOE e c d

RHS



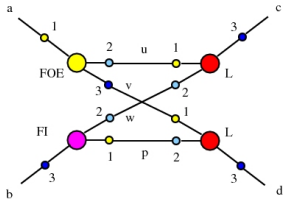
FOE a u v
FI p w b
FO u w c
FO v p d

LHS



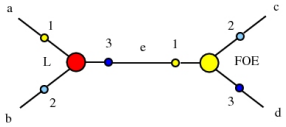
L a b e
FO e c d

RHS



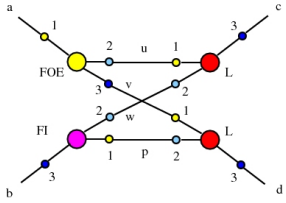
FOE a u v
FI p w b
L u w c
L v p d

LHS



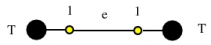
L a b e
FOE e c d

RHS



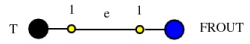
FOE a u v
FI p w b
L u w c
L v p d

LHS



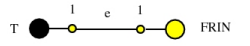
T e
T e

or



T e
FROUT e

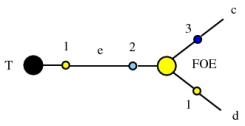
or



T e
FRIN e

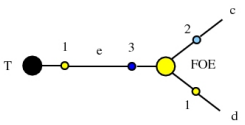
RHS

LHS



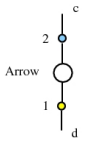
T e
FOE d e c

or



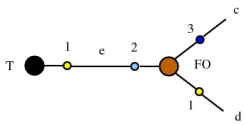
T e
FOE d c e

RHS



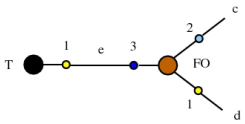
Arrow d c

LHS



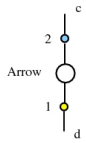
T e
FO d e c

or



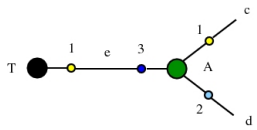
T e
FO d c e

RHS



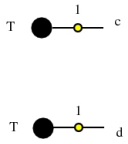
Arrow d c

LHS



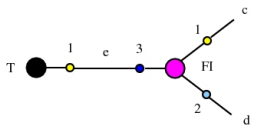
T e
A c d e

RHS



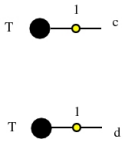
T d
T c

LHS



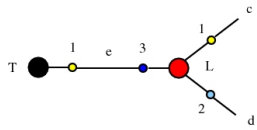
T e
FI c d e

RHS



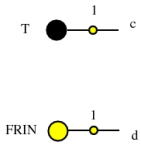
T d
T c

LHS



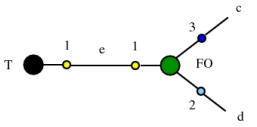
T e
L c d e

RHS



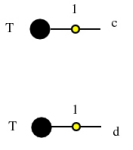
FRIN d
T c

LHS

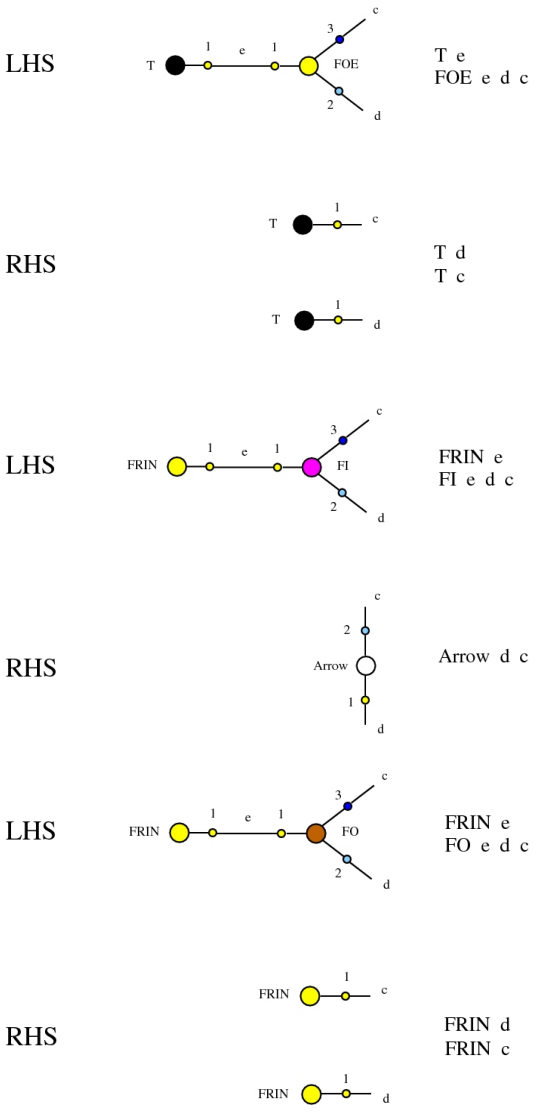


T e
FO e d c

RHS

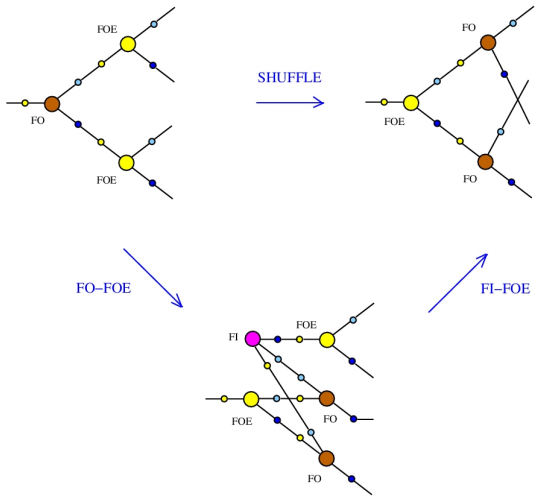


T d
T c



and the COMB rewrites which eliminate Arrow nodes.

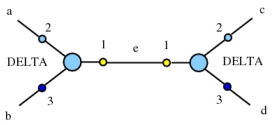
The DISENTANGLE rewrite is no longer needed, because it's effect can be achieved by the SHUFFLE move, which is a sequence of two chemlambda v2 rewrites: FO-FOE and FI-FOE.



4 Interaction Combinators and directed interaction combinators (dirIC)

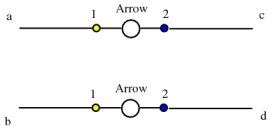
For completeness, here are the graph rewrites of Lafont' Interaction combinators [16], as they appear in the chemlambda v2 js programs, chemistry IC in chemistry.js:

LHS



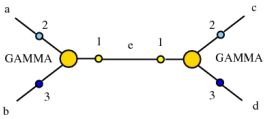
DELTA e a b
DELTA e c d

RHS



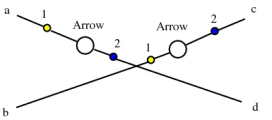
Arrow b d
Arrow a c

LHS



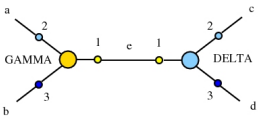
GAMMA e a b
GAMMA e c d

RHS



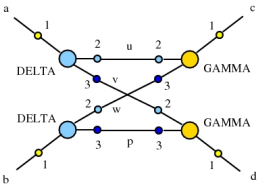
Arrow a d
Arrow b c

LHS



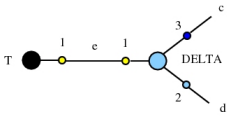
GAMMA e a b
DELTA e c d

RHS



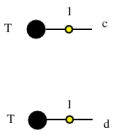
DELTA a u v
DELTA b w p
GAMMA c u w
GAMMA d v p

LHS

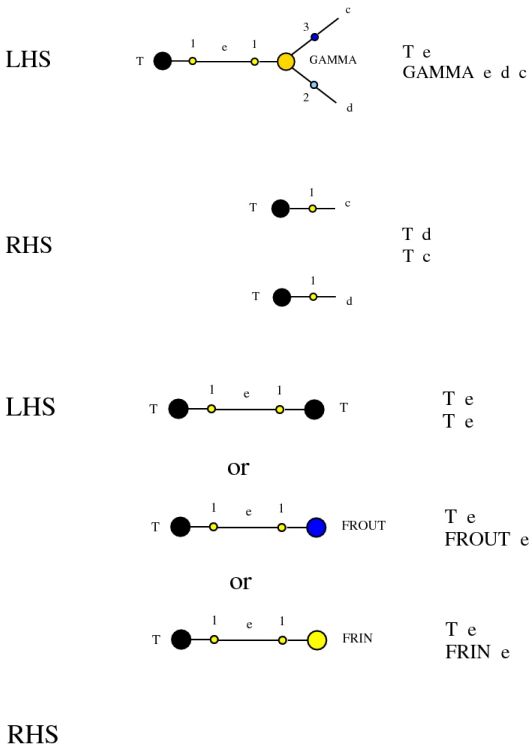


T e
DELTA e d c

RHS



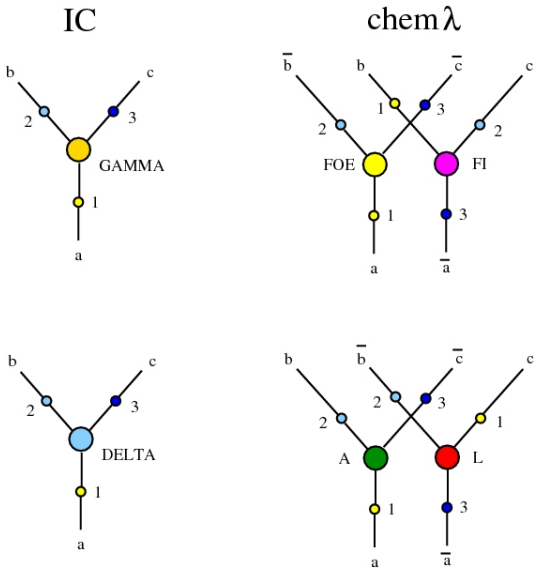
T d
T c



FRIN and FROUT 1-valent nodes are introduced (as in chemlambda) for the free half-edges. We use also the Arrow 2-valent node and the COMB rewrites for eliminating them.

All LHS patterns, of chemlambda and IC, are made of two nodes connected through their active ports. The main difference between chemlambda v2 rewrites and IC rewrites is that several chemlambda nodes have two active ports, while IC nodes have only one active port.

It is possible to modify the chemlambda v2 chemistry such that there are no conflicts. We obtain a chemistry called "dirIC", or directed interaction combinators, explained in [5] Alife properties of directed interaction combinators vs. chemlambda, over the same set of nodes as chemlambda v2. With the chemistry dirIC, there is a translation between IC nodes of Lafont to chemlambda nodes:



As argued here [5], the existence of conflicting rewrites in chemlambda v2, compared

with dirIC, creates much more interesting alife behaviours.

References

[1] A. Bawden, Connection graphs, In: Proceedings of the 1986 ACM conference on LISP and functional programming, 258-265, ACM Press, (1986)

[2] A. Bawden, Implementing Distributed Systems Using Linear Naming, A.I. Technical Report No. 1627, MIT, (1993)

[3] M. Buliga, Graphic lambda calculus, *Complex Systems* **22**, 4, 311-360, (2013)

[4] M. Buliga, Artificial chemistry experiments with chemlambda, lambda calculus, interaction combinators, arXiv:2003.14332, (2020)

[5] M. Buliga, Artificial life properties of directed interaction combinators vs. chemlambda, arXiv:2005.06060 (2020)
https://mbuliga.github.io/quinegraphs/ic-vs-chem.html#icvschem

[6] M. Buliga, Emergent rewrites in knot theory and logic (2020)
https://mbuliga.github.io/novo/presentation.html

[7] M. Buliga, L.H. Kauffman, GLC actors, artificial chemical connectomes, topological issues and knots, arXiv:1312.4333, (2013)

[8] M. Buliga, Lambda calculus to chemlambda. (2019-2020),
https://mbuliga.github.io/quinegraphs/lambda2mol.html#lambdanote

[9] M. Buliga, Chemical concrete machine, Figshare, (2013), arXiv:1309.6914

[10] M. Buliga, L.H. Kauffman, Chemlambda, universality and self-multiplication, ALIFE 2014: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems July 2014, MIT Press, 490-497, (2014)

[11] L.H. Kauffman, Chemlambda, universality and self-multiplication, slides (2014)

[12] M. Buliga, Chemlambda page, https://chemlambda.github.io/index.html (2020)

[13] M. Buliga, Chemlambda project http://chorasimilarity.github.io/chemlambda-gui/index.html (2015-2017)

[14] M. Buliga, Chemlambda-gui repository, Github (2015-2017)
https://github.com/chorasimilarity/chemlambda-gui/blob/gh-pages/dynamic/README.md

[15] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, M. Loewe, Algebraic approaches to graph transformation, part I: Basic concepts and double push-out approach, Handbook of Graph Grammars and Computing by Graph Transformation, 163-245 (1997)

[16] Y. Lafont, Interaction Combinators, *Information and Computation* **137**, 1, 69-101, (1997)

[17] K. Tomita, H. Kurokawa, S. Murata, Graph-rewriting automata as a natural extension of cellular automata. Chapter 14 in: Adaptive Networks, Understanding Complex Systems, T. Gross, H. Sayama (eds.), NECSI Cambridge/Massachusetts (2009)