An Implementation for Framework for Chemical Structure using Graph Grammar

Miss Taniya Seal
PG Student
Department of Computer Science & Engineering
University of Calcutta, India

Samir Kumar Bandyopadhyay
Head of Department
Department of Computer Science & Engineering
Advisor to Chancellor, JIS College, India

Abstract

Modeling molecules as undirected graphs and chemical reactions as graph rewriting operations is a natural and convenient approach to modeling chemistry. Graph grammar rules are most naturally employed to model elementary reactions like merging, splitting, and isomerisation of molecules. In this paper a generic approach for composing graph grammar rules to define a chemically useful rule compositions. We iteratively apply these rule compositions to elementary transformations in order to automatically infer complex transformation patterns.

Keywords: Transformation Rule, Graph Transformation, Graph Grammar

I. INTRODUCTION

Graph grammars, or graph rewriting systems, are proper generalizations of term rewriting systems. A wide variety of formal frameworks have been explored, including several different algebraic ones rooted in category theory. Directed hypergraphs [1] are a suitable topological representation of (bio) chemical reaction networks where (catalytic) reactions are hyper edges connecting substrate nodes to product nodes. Such networks require an underlying Artificial Chemistry [2] that describes how molecules and reactions are modeled. If molecules are treated as edge and vertex labelled graphs, where the vertex labels correspond to atom types and the edge labels denote bond types, then structural change of molecules during chemical reactions can be modeled as graph rewrite [3]. In contrast to many other Artificial Chemistries this approach allows for respecting fundamental rules of chemical transformations like mass conservation, atomic types, and cyclic shifts of electron pairs in reactions. In general, a graph rewrite (rule) transforms a set of substrate graphs into a set of product graphs. Hence the graph rewrite formalism allows not only to delimit an entire chemical universe in an abstract but compact form but also provides a methodology for its explicit construction.

II. REVIEW WORKS

Most methods for the analysis of this network structure are directed towards this graph (or hypergraph) structure [1, 4], which is described by the stoichiometric matrix S of the chemical system. The net reaction of a given pathway is simply the linear combination of the participating hyperedges. In the setting of generative models of chemistry, each concrete reaction is not only associated with its stoichiometry but also with the transformation rule operating on the molecules that are involved in a particular reaction. Importantly, these rules are formulated in terms of reaction mechanisms that readily generalize to large sets of structurally related molecules. It is thus of interest to derive not only the stoichiometric net reaction of a pathway but also the corresponding "effective transformation rule". Instead of attempting to address this issue a posteriori, we focus here on the possibility of composing the elementary rules of chemical transformations to new effective rules that encapsulate entire pathways

III. PROPOSED METHODOLOGY

In the following, the general structure of a graph rewrite rule is explained using the chemical reactions. These can be applied to model chemical reactions based on a graph representation of molecules. Therein, molecules are defined by an undirected graph where each node represents a single atom and edges correspond to bonds of a given chemical structure. As we are modelling chemical reactions, no atoms (i.e. nodes) are allowed to vanish or appear during the reaction. The weight given to an edge describes its bond structure.

A. General Structure of a Rewrite Rule

Each rule is the form of L→R with L being called pattern graph (left hand side) and R being called replacement graph (Right hand side).
In left all edges are specified, which are broken during the chemical transformation. Furthermore, nodes can be listed that change their weights along the reaction, thus they are listed with different weight within the right list. Finally, the matching can be further refined listing other nodes without modifying.

In right all edges are specified, which are formed during the chemical transformation (i.e. which are new in the product molecule(s)). Furthermore, nodes with changing weight (i.e. also listed in left list) are given.

B. General Steps in the Derivation of Writing Rules

It is recommended to follow the protocol below when translating reaction mechanisms into writing rules.
1) Make a sketch of the reaction.
2) Number the atoms in the reaction mechanism.
3) Figure out which atoms/bonds are constant during the chemical transformation.
4) Figure out which bonds are broken during the chemical transformation.(These go into left of the rule).
5) Figure out which bonds are formed during the chemical transformation.(These go into the right of the rule).
6) Check the action of the rule on examples and counter examples to make sure that the rule does what we want.

Here adjacency matrix to store the weight of an edge connecting the nodes. Using the weight matrix I get an advantage from not making a path matrix individually as one element of this matrix gives the information about connected nodes with its bond structure(matrix a[2][3]= 2 means node 2 is connected with node 3 by double bond) .

A good example to illustrate rule is the Cannizzaro reaction. The reaction involves the base-induced disproportionation (i.e. the self oxydation-reduction reaction) of an aldehyde lacking a hydrogen atom in the ȁ-position of the carbonyl-group yielding a 50:50 product mixture of the corresponding alcohol (reduction product) and carboxylic acid (oxydation product). It is shown in figure 1.

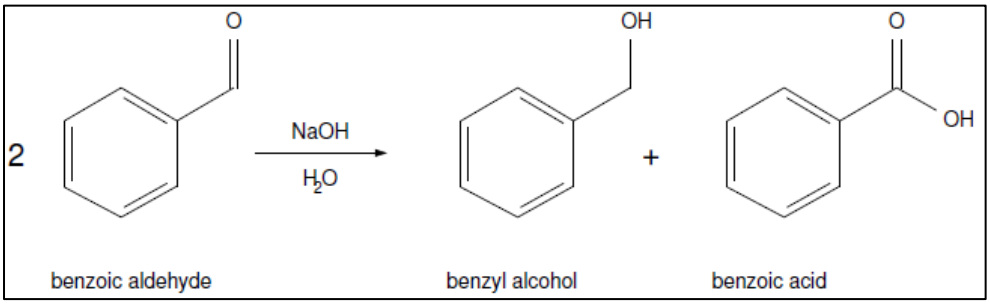

Fig. 1: Cannizzaro Reaction

Let us assume that the canizzarro reaction proceeds via a cyclic six-membered (arranging 2 aldehydes and 1 water molecule) and shown in figure 2

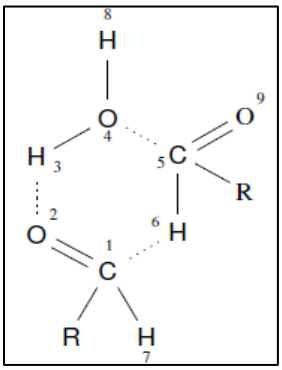

Fig. 2: Cyclic Six-Membered

The following bond changes happen resulting in the rewriting rule.

- Broken bonds: 1-2 (C=O), 3-4 (H-O) and 5-6 (C-H).
- Formed bonds: 1-2 (C-O), 2-3 (H-O), 4-5 (O-C) and 6-1 (H-C).
- Constant: atoms 1-9 and bonds 1-7 (C-H), 4-8 (O-H) and 5-9 (C=O).

The above rule is very general and matches any aldehyde regardless what R actually is. However, aldehydes possessing a hydrogen at the atom adjacent to the carbonyl group (e.g. R = CH3) form the enol tautomer under basic conditions and cannizzaro reaction is not observed. To make the cannizzaro rule specific for aldehydes without a hydrogen in the ȁ position of the carbonyl group, we first have to add two C atoms (10, 11) and the respective bonds (1-10, 5-11) to the context subgraph and disallow hydrogens on atoms 10 and 11.

C. Algoritrhm

Creation of nodes and edges for the input graph.

1) Steps:

1) START

2) Initialize Radius=15, point1=null, point2=null, node1=-1, node2=-1, point2InsideCircle=false.
3) Clicking mouse create node and add to the arraylist named nodeList.
4) For each of the element of the nodeList (arraylist of nodes) do the following:
   4.1) If the distance between clicked point and centre of node <Radius (at the time of mouse pressed event).
   4.1.1) Set node1=i, point1=the centre of the ith element of nodeList.
   4.2) Otherwise set point1=null, node1=-1.
5) For each of the element of the nodeList (arraylist of nodes) do the following:
   5.1) If the distance between clicked point and centre of node <Radius (at the time of mouse released event).
   5.1.1) Set node2=i, point2=the centre of the ith element of nodeList, point2InsideCircle=True.
   5.2) Otherwise set point2=null, node2=-1, point2InsideCircle=False.
6) If point2InsideCircle=True.
   6.1) Enter weight for new edge.
   6.2) If weight =Null.
   6.2.1) Make edge with point1 and point2 and weight=0.0.
   6.3) Otherwise make edge with point1 and point2 and weight given by user.
   6.4) if Arraylist edgeList contains the new edge.
   6.4.1) Print edge already exist.
   6.5) Otherwise add the new edge in edgeList (arraylist of edges).
7) Otherwise print point2 is not properly entered.
8) END.

Apply the rule of graph grammar

2) Steps

1) START
2) Make a frame named Graph input and Make a button named "Generate matrix".
3) Click the Button.
4) For each of the element of the nodeList (arraylist of nodes) do the following:
   4.1) Array a[node1 of i th element of edgeList][node2 of i th element of edgeList]=weight of i th element of edgeList.
5) For each row of array a do the following:
   5.1) For each column of array a do the following:
   5.1.1) Print elements of adjacency matrix a[i][j] of given graph.
6) If a[0][1]=2 and a[2][3]=1 and a[4][5]=1(left side of the rule)
   6.1) a[0][1]=1, a[1][2]=1, a[3][4]=1,a[0][5]=1,a[2][3]=0,a[4][5]=0(update the weight after chemical transformation. Weight '0' means bond breaks between the atoms. This is the right hand side of the rule).
7) For each row of the array a do the following:
   7.1) For each column of the array a do the following:
   7.1.1) Print elements of adjacency matrix a[i][j] of updated graph.
8) If a[4][8]=2 and a[4][9]=1 and a[3][4]=1 & a[3][7]=1(matching)
   8.1) Print carboxylic acid is found.
   8.2) Create graph for carboxylic acid.
9) Otherwise continue with 10.
10) 10. If a[0][1]=1 and a[1][2]=1 and a[0][5]=1 and a[0][6]=1 and a[0][10]=1
   9.1) Print alchohol is found.
   9.2) Create graph for alchohol .
11) Otherwise Print No structure is matched.
12) END.

Construct graph for chemical structure after transformation

3) Steps

1) START
2) Make a frame named Graph Ouput.
3) Create the scene with default viewport.
4) Create some graphic objects.
5) Connect the objects by a line, draw circle for objects and maintain the hierarchy (mentioning the parent object).
6) END.

D. Methods Used

Table - 1

CLASS NAME	***METHOD NAME***
GraphGUIInput	***main()***
	GraphPanel()

GraphPanel	***paint(graphic gr)***
GraphListener	***mousePressed()*** ***mousedragged()*** ***mouseReleased()***
Node	***Node()***
Edge	***Edge()*** ***hashCode()*** ***equals()***
Result	***Result()*** ***event(Gscene,int,int,int)*** ***draw()***
TestObject	***TestObject(String, GObject,double ,double)***
Result1	***Result1()*** ***event(Gscene,int,int,int)*** ***draw()***

IV. RESULT

Screenshots of Successful formation of chemical products are shown in Figure 6.

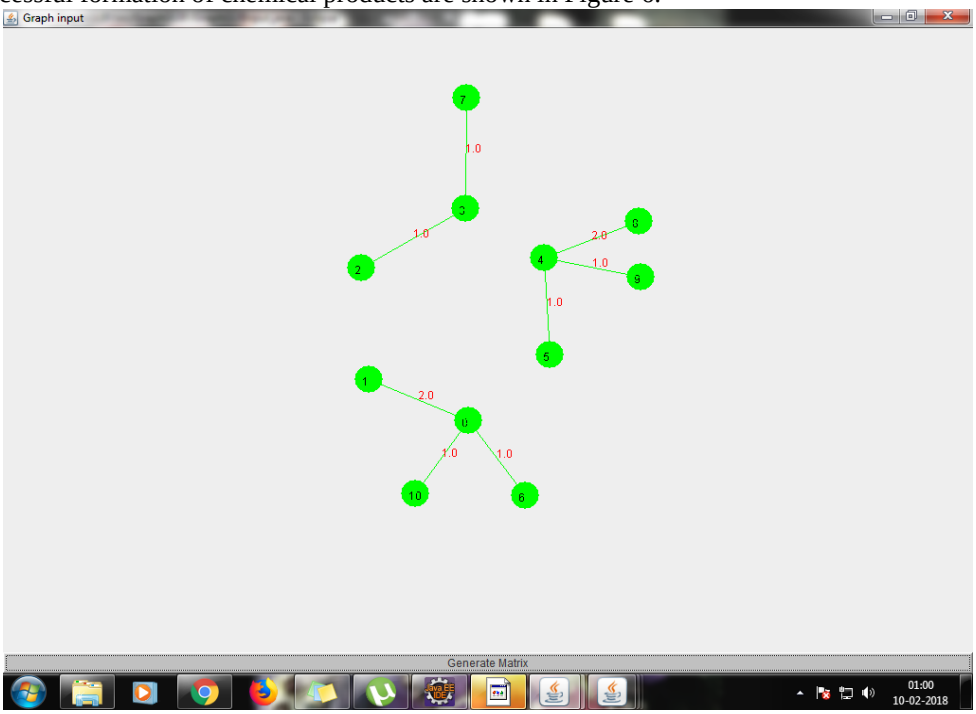

Fig. 6: Formation of Chemical Product

This figure 6 shows nodes and edges by clicking the mouse and dragging the mouse from one node to another respectively by Java GUI. Weight is given for every edge when making an edge.

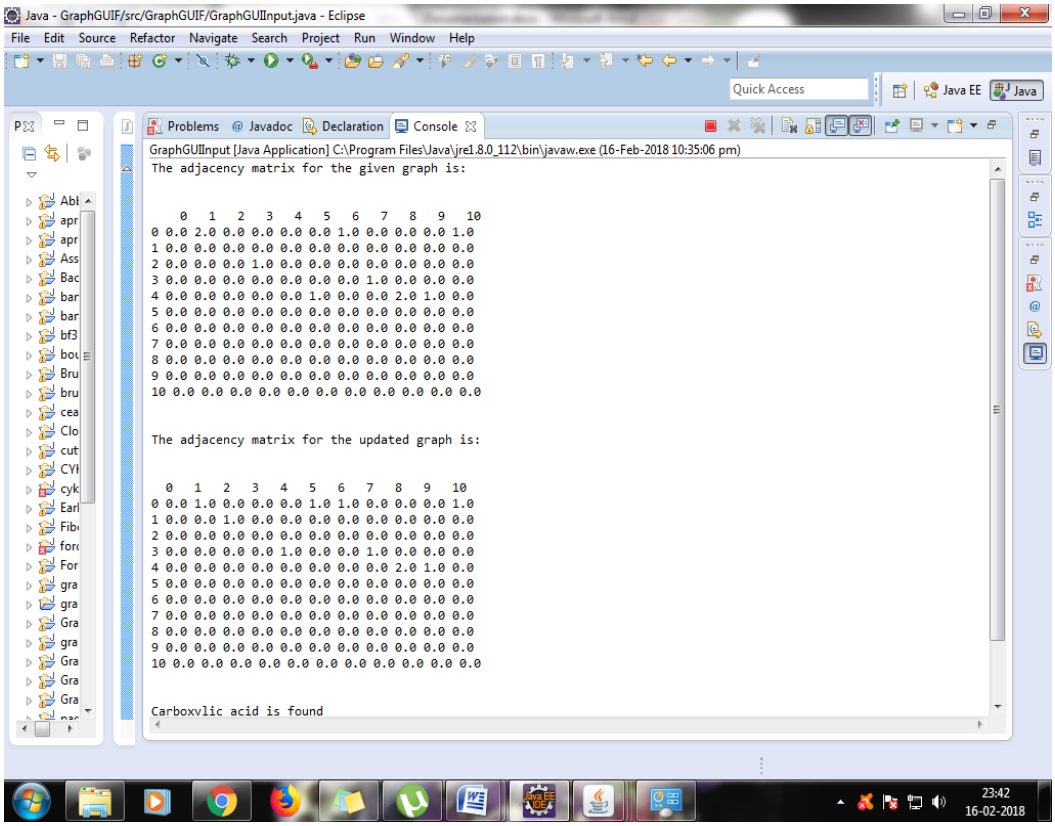

Fig. 7: Adjacency Matrix

By clicking the button "Generate matrix". The adjacency matrix is generated in console output. After applying the rule the adjacency matrix is updated by its weight and also is printed in console.

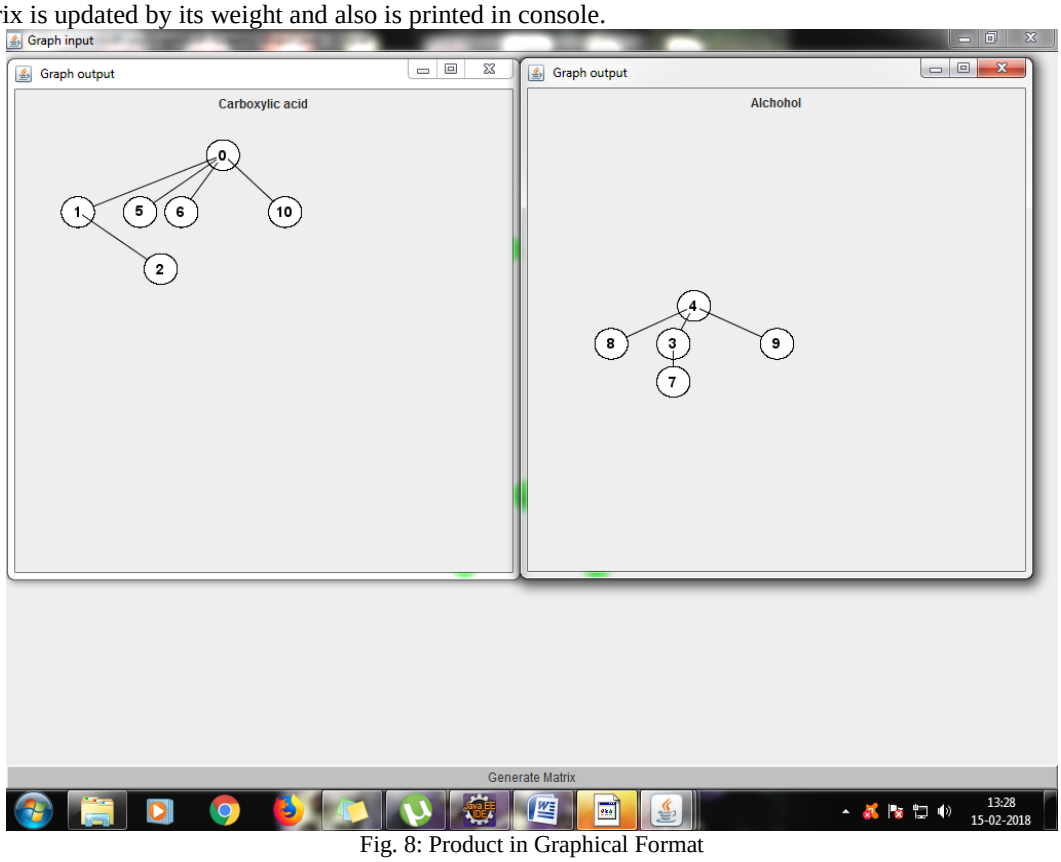

Fig. 8: Product in Graphical Format

After matching with products formed in chemical transformation, two windows are generated where the connections of products are shown in graph after applying the rule. Here the use Java 2D Graphics Library is used to represent the products in graphical format. It is shown in Figure 7.

A. Features of this Library

- Object oriented hierarchical scene graphs
- Layered graphics with visibility support
- Flexible 3D space world extents
- World and device coordinate support
- Powerful rendering style support
- Smart annotations
- Powerful object detection functions
- Extensible
- Interaction support
- Raster image support
- Embedded Swing component support
- Utilities for geometry generation and transformations
- Image export and printing support
- JDK 1.2, 1.3, 1.4 and 1.5 compatible
- Light weight (~80kB), self-contained, simple to use.

Screenshots of Unsuccessful formation of chemical products are shown in the following figures.

1) Case 1

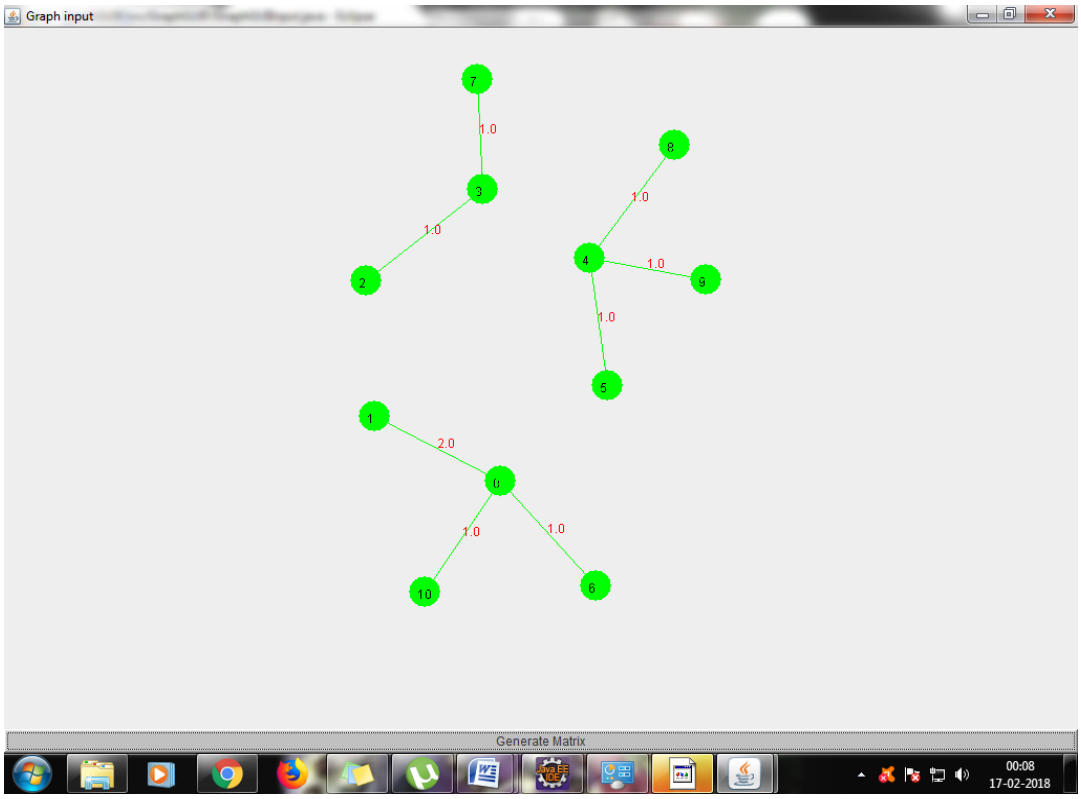

Fig. 9:

Here the atoms 4,5,8,9 do not represent an aldehyde as the bond between 4 and 8 is not a double bond. So in reaction with water molecule (2, 3, 7 atoms) Carboxylic acid cannot be formed.

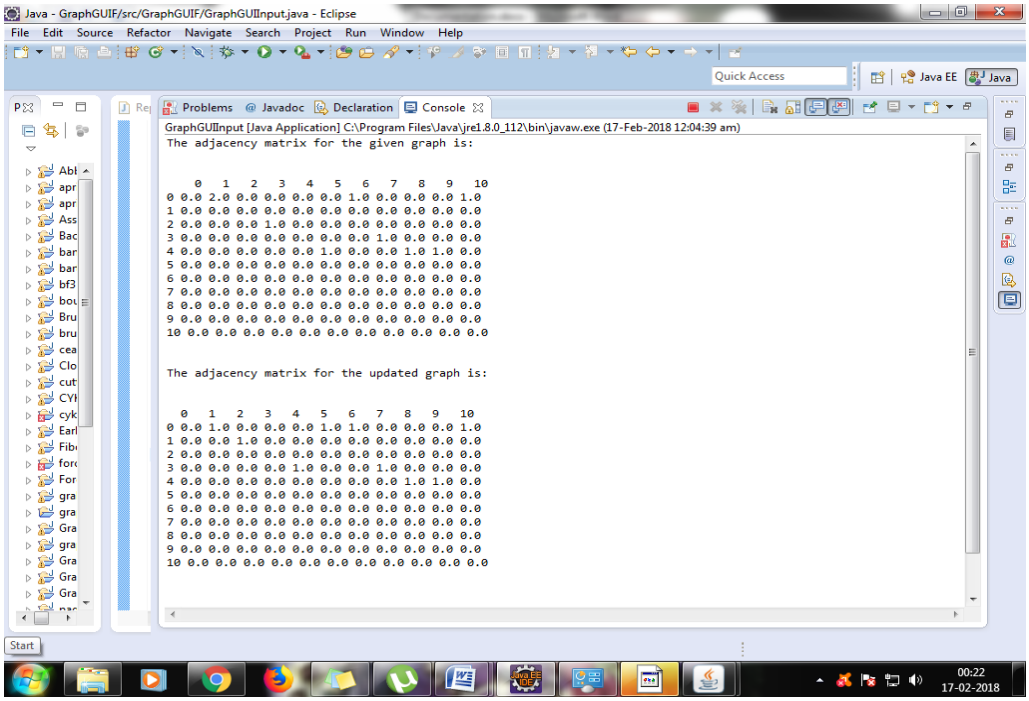

Fig. 10:

Here is the adjacency matrix.

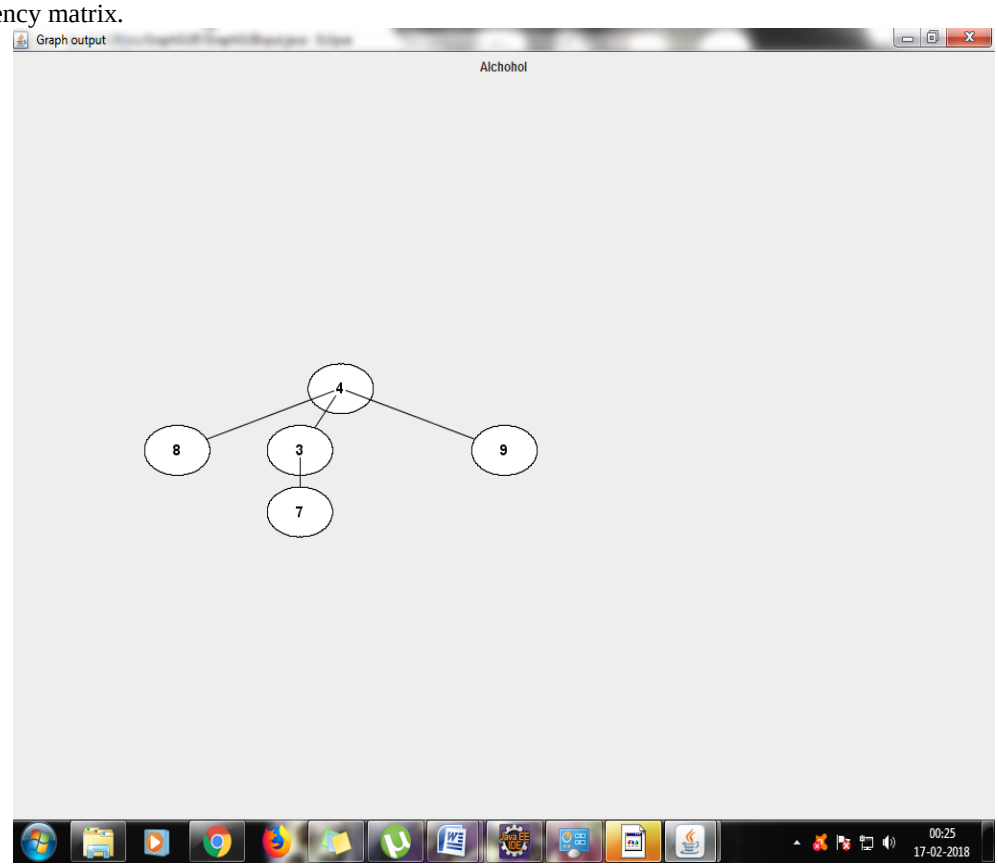

Fig. 11:

So only Alcohol is formed.

2) Case 2

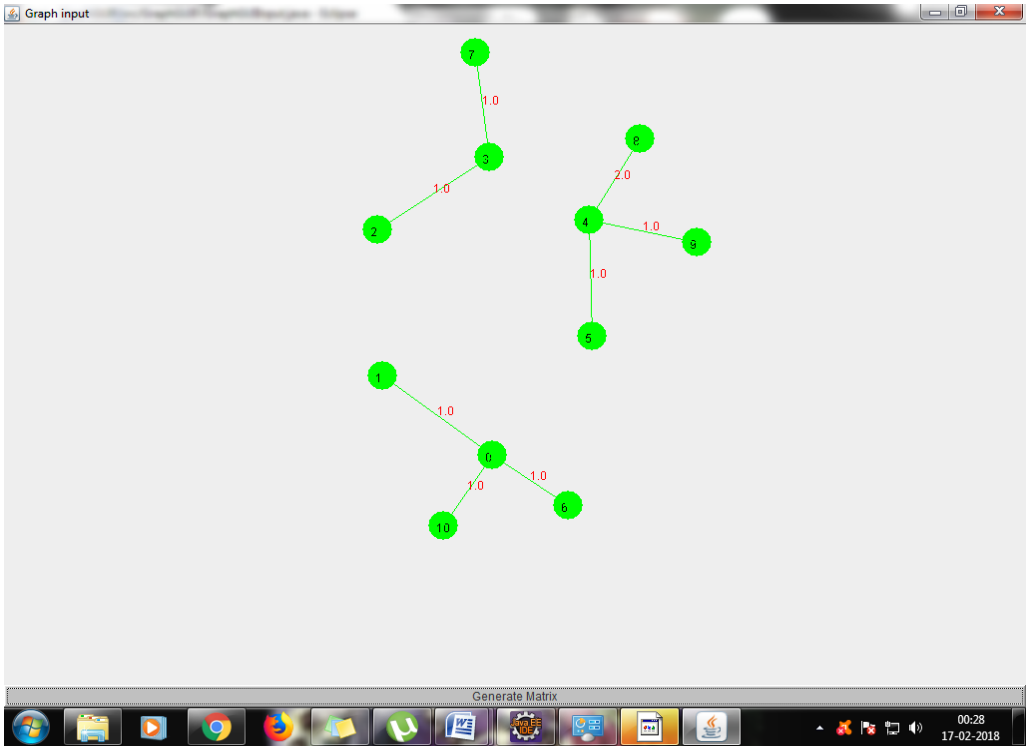

Fig. 12:

Here the atoms 0, 1, 6, 10 do not represent an aldehyde as the bond between 0 and 1 is not a double bond. As it does not follow the rule, No chemical structure is generated.

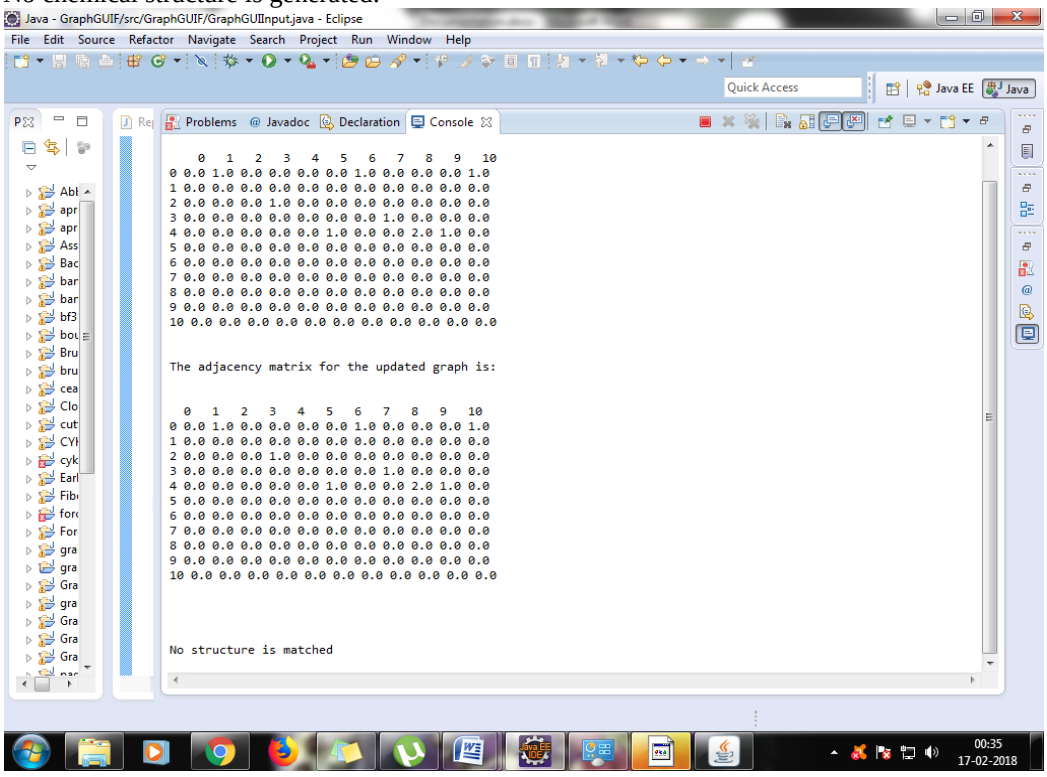

Fig. 13:

Here is the adjacency matrix.

V. CONCLUSIONS

We introduce a generic approach for composing graph grammar rules to define a chemically useful rule compositions. We iteratively apply these rule compositions to elementary transformations in order to automatically infer complex transformation patterns.

REFERENCES

[1] Klamt S, Haus UU, Theis F: Hypergraphs and cellular networks. PLoS Comput Biol 2009, 5(5):e1000385.
[2] Dittrich P, Ziegler J, Banzhaf W: Artificial chemistries – a review. Artif Life 2001, 7(3):225–275.
[3] Kanehisa M, Goto S, Sato Y, Furumichi M, Tanabe M: KEGG for integration and interpretation of large-scale molecular data sets Nucleic Acids Res 2012, 40:D109–114.
[4] Mann M, Ekker H, Flamm C: The graph grammar library – a generic framework for chemical graph rewrite systems. 2013.