International Journal of Emerging Engineering Research and Technology
Volume 2, Issue 9, December 2014, PP 63-69
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Metabolic Pathway Analysis Using Constraint Programming

Howida Ali Abd Elgader
Sudan University of Science & Technology
Hoida_1@hotmail.com

Mohamed Aboualhoda
Nile University Egypt
Mabouelhoda@yahoo.com

Abstract: *A pathway is a set of interactions, or functional relationships, between the physical and/or genetic components of the cell which operate in concert to carry out a biological process. These interactions derive a huge graph with a high degree of connectivity, including artifacts and false-positive prediction, which is difficult to interpret. The prototype of the system is written in Prolog and produced an analysis as Prolog facts, so the interpretation is accessible for answering a number of biological queries about the pathway. Features of prototype are illustrated by case study on the Glyconlysis/Gluconeogensis- pathway*

1. INTRODUCTION

Systems biology means understanding the structure and dynamics of cellular and organism functions. It means to investigate how genes and cells interact and react to stimuli from the environment. In order to gain understanding of complex biological systems, the science of biology cannot stand alone. The roles of other sciences are important, such as chemistry, engineering, computer science, mathematics, and physics. Engineering and computer science play a significant role in systems biology as they help in providing modeling tools, visualization and simulation, data exchange formats, and databases [1].

A pathway is a set of interactions, or functional relationships, between the physical and/or genetic components of the cell which operate in concert to carry out a biological process. Metabolic pathway databases generally contain detailed data models that represent a pathway as a series of biochemical reactions, focusing mainly on the chemical modifications made to the small molecule substrates of enzymes. Many metabolic pathways have been mapped to the molecular level of detail since the 1950s or earlier and metabolic pathway databases are the earliest and perhaps the best-known. Metabolic databases generally do not represent higher order cellular processes, such as gene regulation [2].

Kyoto Encyclopedia of Genes and Genomes (KEGG) contains information about molecular interactions networks namely, metabolic pathway, regulatory pathways, and molecular assemblies. It also contains gene catalogue or all the organisms that have been sequenced and links each gene product to a component on the pathway. Therefore all chemical compounds in living cells and links of each compound to a pathway component are found in KEGG. It consists of pathways and complexes (PATHWAY database), genes and proteins (GENES/SSDB/KO databases), information about chemical compounds and reactions (COMPOUND / DRUG / GLYCAN / REACTION database) [3]. There is a strong emphasis in the KEGG project on the graphical presentation of biochemical pathways. This emphasis is reflected in KGML (KEGG Markup Language), in which tags correspond to features of the graphical layout. KEGG also provided the specification for PATHWAY database, which consists of metabolic pathways and Regulatory pathways, in XML DTD format (KEGG DTD) and the data in XML format.

These interactions derive a huge graph with a high degree of connectivity, including artifacts and false-positive prediction, which is difficult to interpret. So an environment is needed for answering queries about the graph, or a method capable of structurally representing the pathway in a form that can be readily processed by computer and easily understood by humans would be of great value.

Constraint Logic Programming (CLP) represents a successful attempt to merge the best features of logic programming (LP) and constraint solving. Much knowledge about a domain can be captured in logical expressions. Logic programming provides a means to compute new information based on captured information. Prolog is a programming language that carries computation over logical expressions. Logic programs have two types of logical expressions: facts and rules. Facts are atomic

expressions formed from a predicate with arguments -- the arguments contain terms. A term is either a variable, a constant, or a function applied to a term. A rule has a body and a head: Head ~ Body. It expresses the notion that Head is true if Body is true. The head of a rule is an atom. The body of a rule is a (possibly empty) conjunction of atoms. A rule with an empty body is a fact. Constraint solving includes a variety of expressive modeling frameworks and efficient solving tools for real-life problems that can be described via a set of variables and constraints over them. A constraint is just a restriction imposed over the combination of values of some variables of the problem. Solving a problem with constraints means finding a way to assign values to all its variables such that all constraints are satisfied. Constraint solving methods have been successfully applied to many application domains, such as scheduling, planning, resource allocation, vehicle routing, computer networks, and bioinformatics [4].

2. THE METHOD

The prototype of the system is written in Prolog and produced an analysis as Prolog facts, so the interpretation is accessible for answering queries about the pathway. Another necessary component is encoded knowledge about metabolic pathways. The KEGG database provides this information. The pathway from KEGG has been parsed into Prolog facts as list of connected nodes, and other supporting information. Each object-attribute structure is encoded as one or more Prolog facts. Prolog rule capture operator over the objects. Together the rules and facts form a knowledge base that can be used to answer queries.

3. FEATURES OF KEGG PATHWAY

The KEGG pathway maps are graphical image maps representing networks of interacting molecules responsible for specific cellular functions. There are two types of KEGG pathways:

- Reference pathways which are manually drawn and
- Organism-specific pathways which are computationally generated based on reference pathways.

The KGML files contain computerized information about graphical objects and their relations in the KEGG pathways as well as information about orthologous gene assignments in the KEGG GENES database.

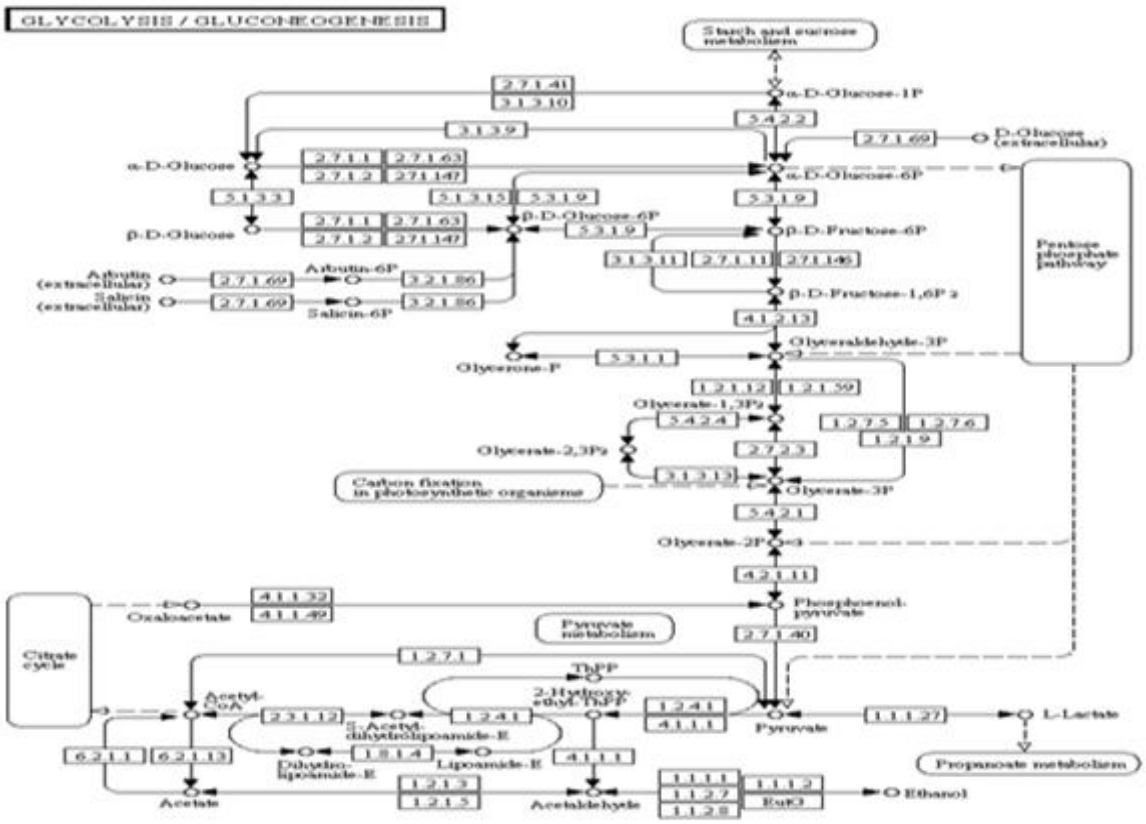

Fig1. *Example of KEGG pathway map [5]*

In KGML the **pathway** element specifies one graph object with the **entry** elements as its nodes and the **relation** and **reaction** elements as its edges. The relation and reaction elements indicate the connection patterns of rectangles (gene products) and the connection patterns of circles (chemical compounds), respectively, in the KEGG pathways. The two types of graph objects, those consisting of entry and relation elements and those consisting of entry and reaction elements, are called the protein network and the chemical network, respectively. Since the metabolic pathway can be viewed both as a network of proteins (enzymes) and as a network of chemical compounds, another distinction of KEGG pathways is:

- Metabolic pathways viewed as both protein networks and chemical networks and
- Regulatory pathways viewed as protein networks only [5].

The KEGG Markup Language (KGML) is an exchange format of the KEGG graph objects, especially the KEGG pathway maps that are manually drawn and updated. KGML enables automatic drawing of KEGG pathways and provides facilities for computational analysis and modeling of protein networks and chemical networks.

```
<?xml version="1.0" ?>
 <!DOCTYPE pathway (View Source for full doctype...)>
- <!-- Creation date: Sep 30, 2010 09:16:33 +0900 (GMT+09:00)
 -->
- <pathway name="path:ec00010" org="ec" number="00010" title="Glycolysis / Gluconeogenesis" image="http://www.genome.jp/kegg/pathway/ec/ec00010.png" link="http://www.genome.jp/kegg-bin/show_pathway?ec00010">
- <entry id="13" name="ec:4.1.2.13" type="enzyme" reaction="rn:R01070" link="http://www.kegg.jp/dbget-bin/www_bget?4.1.2.13">
 <graphics name="4.1.2.13" fgcolor="#000000" bgcolor="#BFBFFF" type="rectangle" x="483" y="404" width="46" height="17" />
 </entry>
- <entry id="37" name="ec:1.2.1.3" type="enzyme" reaction="rn:R00710" link="http://www.kegg.jp/dbget-bin/www_bget?1.2.1.3">
 <graphics name="1.2.1.3" fgcolor="#000000" bgcolor="#BFBFFF" type="rectangle" x="289" y="943" width="46" height="17" />
 </entry>
- <entry id="38" name="ec:6.2.1.13" type="enzyme" reaction="rn:R00229" link="http://www.kegg.jp/dbget-bin/www_bget?6.2.1.13">
 <graphics name="6.2.1.13" fgcolor="#000000" bgcolor="#BFBFFF" type="rectangle" x="146" y="911" width="46" height="17" />
 </entry>
- <entry id="39" name="ec:1.2.1.5" type="enzyme" reaction="rn:R00711" link="http://www.kegg.jp/dbget-bin/www_bget?1.2.1.5">
 <graphics name="1.2.1.5" fgcolor="#000000" bgcolor="#BFBFFF" type="rectangle" x="289" y="964" width="46" height="17" />
 </entry>
- <entry id="40" name="cpd:C00033" type="compound" link="http://www.kegg.jp/dbget-bin/www_bget?C00033">
 <graphics name="C00033" fgcolor="#000000" bgcolor="#FFFFFF" type="circle" x="146" y="953" width="8" height="8" />
```

Fig2. *A part of KEGG pathway in KGML format [5]*

4. DATA EXCHANGE FORMAT

An XML parser written in Prolog is used to parse the XML data into a Prolog term structure. Figure 2 illustrates the structure of data in a KEGG XML document and the corresponding Prolog term structure representation is illustrated in Figure 3.

An XML parser written in Prolog is used to parse the XML data into a Prolog term structure.

```
:-use_module(library(sgml)).
clear_all:-retractall(entry),
        retractall(relation),
        retractall(reaction),
        retractall(subtype),
        retractall(product),
        retractall(substrate),!.
clear_all.
startparsing:-clear_all,load_xml_file('d:/pathway.xml',XML),
          XML=[element(A1,A2,Kids0)],
          assert(pathway(A1,A2)),
          nodes(Kids0).
nodes([_]):-!.
nodes([_,element(X1,X2,X3)|T]):-
        convertarg(X2,[],IX2),
        reverse(IX2,[],X4),
        createfact(X1,X4,X3),
        nodes(T),!.
onefact([_,element(X1,X2,_)|T],F1,F2):-
        convertarg(X2,[],IX2),
        reverse(IX2,[],X4),
        F1=..[X1,X4],anotherfact(T,F2).
anotherfact([_,element(X1,X2,_)|_],F1):-
        convertarg(X2,[],IX2),
        reverse(IX2,[],X4),
        F1=..[X1,X4].
createfact(entry,[A1,A2,A3,A4],_):-assert(entry(A1,A2,A3,A4)).
createfact(entry,[A1,A2,A3,A4,A5],_):-assert(entry(A1,A2,A3,A4,A5)).
createfact(relation,[A1,A2,A3],Kids1):-assert(relation(A1,A2,A3)),nodes(Kids1).
createfact(reaction,[A1,A2,A3],Kids1):-onefact(Kids1,A4,A5),assert(reaction(A1,A2,A3,A4,A5)).
createfact(subtype,[A1,A2],_):-assert(subtype(A1,A2)).
createfact(product,[A1,A2],_):-assert(product(A1,A2)).
createfact(substrate,[A1,A2],_):-assert(substrate(A1,A2)).
```

```
convertarg([],X,X):-!.
convertarg([X|T],Y,A):-arg(2,X,Q),
                atom_length(Q,L),
                P is L -1,
                getvalue(Q,0,P,Z),
                convertarg(T,[Z|Y],A).
reverse([],List,List).
reverse([X|Tail],SoFar,List) :-
reverse(Tail,[X|SoFar],List).
getvalue(T,X,X,A):-atom_number(T,A),!.
getvalue(T,Z,V,Q):-sub_atom(T,Z,1,_,S),
              atom_to_chars(S,B),
               B=[M],
               between(48,57,M),
               U is Z+1 ,
               getvalue(T,U,V,Q),!.
getvalue(A,_,_,C):-C=A.
```

Fig3. *An XML parser written in Prolog*

Facts containing common names of specific pathway have been parsed from KEGG into the form:

```
pathway(name,org,number,title,image,link)
entry(id,name,type,link,reactiom)or entry(id,name,type,link)
reaction(id,name,type,substrate(id,name),product(id,name))
relation(entry1,entry2,type,subtype(name,val))
```

Fig4. *Facts containing common names*

5. BIOLOGICAL QUERIES

Below we enumerate a list of some biological queries of interest:

- *Path search*

Can infer all the paths from the source node to the target node, and can give all information about these nodes.

```
:-consult('D:/metap/parser.pl').
:-startparsing.
 neighborroom(Source_Id,Destination_Id):-relation(Source_Id ,Destination_Id,_).
 neighborroom(Source_Id,Destination_Id):-relation(Destination_Id,Source_Id,_).
  route(Room,Room,VisitedRooms):-
       member(Room,VisitedRooms),
       write(VisitedRooms),nl,nodedes(VisitedRooms).
  route(Room,Way_out,VisitedRooms):-
       neighborroom(Room,NextRoom),
```

```
    \+(member(NextRoom,VisitedRooms)),
    route(NextRoom,Way_out,[NextRoom|VisitedRooms]).
member(X,[X|_]).
member(X,[_|H]):-member(X,H).
nodedes([]).
nodedes([N|X]):-entry(D1,N ,D2 ,D3 ,_),write('node '),write(N),
        write(' is: '),write(D1),write(','),
        write(D2),write(','),write(D3),
        nl,nodedes(X).
go(Here,There):-route(Here,There,[Here]).
go(_,_).
```

Fig5. *Code of infer all the paths from the source node to the target node*

Case Study

When we use Glyconlysis/Gluconeogensis- pathway some results are follow:

- *Path search*

go (**59**,**79**)

[59,13,58,56,77,57,55,54,52,50,41,67,70]

[59,62,60,13,56,77,57,55,54,52,50,41,67,70]

[59,64,60,13,56,77,57,55,54,52,50,41,67,70]

[59,13,56,77,57,55,54,52,50,41,67,70]

[59,62,60,13,58,56,77,55,54,52,47,50,41,67,70]

[59,64,60,13,58,56,77,55,54,52,47,50,41,67,70]

- *Node Information*

?- entry(64,Name,-,-).

Name=’ec:5.3.1.9’

- *Reaction information*

?- reaction(82,Name,Type,Z,V).

Name=’rn:R05133’

Type=irreversible

Z=substrate([107,’cpd:c06187’])

V=product([90,’cpd:c01172’])

6. CONCLUSION

Engineering and computer science play a significant role in systems biology as they help in providing modeling tools, visualization and simulation, data exchange formats, and databases. A pathway is a set of interactions which derive a huge graph with high degree of connectivity which is difficult to interpret. So an environment is needed for answering queries about the graph, or a method capable of structurally representing the pathway in a form that can be readily processed by computer and easily understood by humans would be of great value. In this paper we have proposed constraint logic programming to solve this problem. Constraint Logic Programming (CLP) represents a successful attempt to merge the best features of logic programming (LP) and constraint solving. The prototype of the system is written in Prolog and produced an analysis as Prolog facts, so the interpretation is

accessible for answering queries about the pathway. After testing the results were 100% correct and in record time

REFERENCES

[1] Selpi. (2004) an FDM Prototype for Pathway and Protein Interaction Data. Chalmers University Of Technology Goteborg Sweden

[2] Michael P. Cary, Gary D. Bader, Chris Sander (2005) Minireview Pathway Information for Systems Biology. Computational Biology Center, Memorial Sloan-Kettering Cancer Center New York, USA

[3] Orpita Bosu, Simminder Kaur Thukral (2007) Bioinformatics Databases, Tools and Algorithms. Oxford University Press

[4] Marco Gavanelli and Francesca Rossi Constraint Logic Programming. Springer-Verlag Berlin Heidelberg 2010

[5] KEGG Markup Language .http://www.genome.jp/keg/xml/. Down loud 19/07/2010

[6] Terry Gaasterland, Evgeni Selkov. Reconstruction of Metabolic Networks Using Incomplete Information

[7] Ken-ichiro Fukuda and Toshihisa Takagi (2001). Knowledge Representation of Signal Transduction Pathway. Oxford University Press 2001

[8] Nathalie Chabrier, Marc Chiaverini, Vincent Danos , Francois Fages and Vincent Schachter (2004) Modeling and Querying Molecular Interaction Networks

[9] John E.Beasley and Francisco J. Planes Recovering Metabolic Pathway via Optimisation. Published by Oxford University Press 2006

[10] Peter D. Karp, Monica Riley Representation of Metabolic Knowledge. First International Conference on Intelligent Systems for Molecular Biology, Bethesda, MD, 1993, Morgan Kaufmann Publishers, pp207-215.