IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-3, March 2019 E-ISSN: 2347-2693

Generation of Paths and Cycles Using Hyperedge Replacement and Their Learning

Thanga Murugeshwari. V[1*], Emerald Princess Sheela J.D.[2]

[1,2]Department of Mathematics, Queen Mary's College, Chennai, Tamil Nadu, India

[*]*Corresponding Author:thangammathe05@gmail.com, Tel.: 9500402807*

DOI: https://doi.org/10.26438/ijcse/v7i3.326330 | Available online at: www.ijcseonline.org

Accepted: 23/Mar/2019, Published: 31/Mar/2019

***Abstract*—** In this paper, we generate paths and cycles using hyperedge replacement graph grammars and hyperedge replacement graph P systems. We observe that the generative power is increased when we use P system to generate paths and cycles. This paper is the impact of Jeltsch and Kreowski work on grammatical inference based on hyperedge replacement. For special classes of graphs namely paths and cycles an alternative method is given to infer the exact grammar using edge contraction between the adjacent vertices.

***Keywords*—** Graph Grammars, Hyperedge replacement, Grammatical Inference**.**

I. INTRODUCTION

In 1990, D. Janssens and G. Rozenberg introduced Graph Grammars. For Multi-dimensional datas, structural manipulations are described by parsing and generating graphs using graph grammars. Grammatical manipulation of Graphs are used in processing of multi-dimensional pattern classes which is often difficult to describe in string Grammars. Hence in the context of languages, graph grammar is a graph rewriting system. The production rule of a graph grammar consists of the mother graph ,daughter graph and an embedding mechanism. In a host graph, the production rule can be applied by removing the mother graph and joining the daughter graph by using the embedding mechanism [1 2] . From the large number of context-free graph grammars that have been investigated, the hyperedge replacement, node replacement, and edge replacement grammars are the three types of context free graph grammars that are mostly used [3].

A P system consists of membranes that are arranged hierarchically inside the skin membrane. The two main components of the membrane are objects and evolution rules. The rules transforms the object from one membrane to the other or it can also dissolve the membrane where the object is placed. P systems acts as a bridge between well-known models of computations in mathematics, computer science and biology[4].

The field of grammatical inference is applied to a number of research areas including machine learning, formal language theory, computational linguistics, syntactic pattern recognition, and biology [5]. Path and Cycle graphs have applications in combinatorics. Existence of Hamiltonian Path and Cycle leads to powerful results in Graph Theory and further it has numerous *applications* in various fields such as networking, block designs, and bioinformatics.

In section II, basic definitions with respect to hyperedge replacement graph grammars and P systems are recalled. In section III , generation of paths and cycles is defined using HRG followed by generation paths and cycles is defined using P system in section IV.In section V, learning algorithm is giver to infer paths and cycles from a given sample set..

II. PRELIMINARIES

Definition 2.1: **[2]** Let K be an arbitrary but finite set of labels and let type be a typing function from K to ℕ. A ***Hypergraph*** H over K is a tuple (Vertices , Hyperedges, attach, labelling, external) Where attach : Hyperedges →Vertices* is a mapping that assigns a sequence of pair wise distinct attachment node attach(e) to each hyperedge e, labelling is a mapping from Hyperedges to K that labels each hyperedge such that type(labelling(e)) = |attach(e)|, external ∈ Vertices* is a sequence of pair wise distinct external nodes. H_k denotes set of all hypergraphs over K

Definition 2.2:[2] A ***hyperedge replacement grammar*** is a system HRG= (NT, TE, PR, S), Where NE ⊆K is a set of non-terminals. TE ⊆ K with TE ∩ NE = ϕ is a set of terminals, PR is a finite set of productions. A production over NE is an ordered pair PR = (A, R) with A ∈ NE, R ∈ H_c and

© 2019, IJCSE All Rights Reserved

type (A) = type(R). A is called the left-hand side of P and is denoted by lhs (PR). R is called the right-hand side and is denoted rhs(PR). S ϵ N is a start symbol.The class of all hyperedge replacement grammars is denoted by HRG .

Definition 2.3:[2]A ***m-hypergraph*** is defined as a hypergraph with m external nodes and a handle e (single hyperedge) with attach H(e) = ext H. If labelling H(e) = A, then H is said to be handle induced by A and is denoted by $A^{\bullet}$.

Definition 2.4:[2] The ***hypergraph language*** L(HRG) generated by HRG is L_s(HRG) where for all A ϵ NE. L_A(HRG) consists of all hypergraphs in H_T derivable from $A^{\bullet}$ by applying productions of P . We denote the class of all hyperedge replacement grammars by HRL.

The concepts of ***hyperedge replacement graph P system with conditional communication*** and ***maximal parallelism rewriting step*** has been discussed in [6,7] respectively.

III. GENERATION OF PATHS AND CYCLES USING HRG

Consider the hyperedge replacement grammar HRG = {{D, K}, {p_1, p_2, p_3}, z} where the axiom and the three productions are given in Figure 1

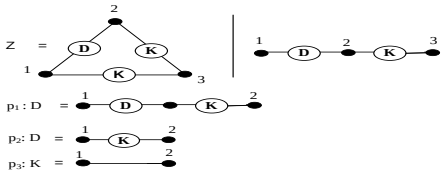
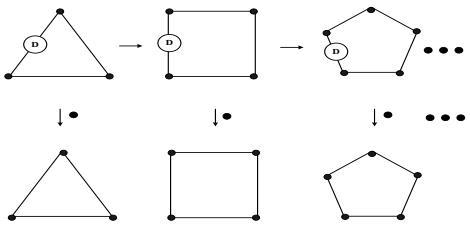


Figure 1 Generation of Cycles and Paths

The axiom contains a hyperedge D, K. If p_1 is applied, the D-edge is replaced by another path of length two. If p_2 is applied the D edge is replaced by another path which erases the D-edge such that neither p_1 nor p_2 can be applied, Further K-edges can be replaced by ordinary edges, eventually. In axiom z, the edge (1,3) and (2,3) with non-terminal K will be replaced by the edge (1,2) of p_3. In p_1 the unlabelled vertex is now labelled 2. The following figure illustrates the generation of cycles C_n and paths P_n from the above production rules. Then, L (HRG) = {C_n/n ≥3} ∪ {P_n/n≥3}

Generation of Cycles:



Generation of Path:

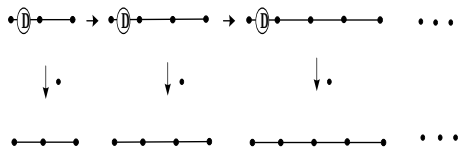


Figure 2 Generation of Cycles and Paths

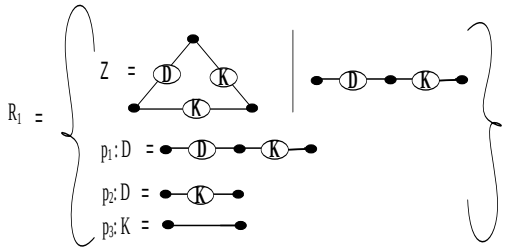
IV. GENERATION OF PATHS AND CYCLES USING HYPEREDGE REPLACEMENT GRAPH P SYSTEM WITH CONDITIONAL COMMUNICATION

Hyperedge replacement graph P system that generates paths and cycles are given below. Initially we construct a P system with a single membrane and later we construct a P system using double membrane with maximal parallelism in which we have two initial non terminals. Maximum parallelity in double membrane increases the generative power of this language.

4.1 Generation of Paths and Cycles in single membrane

Π=(V,T,μ,M_1, ,(R_1,P_1,F_1),(1, 1))

V={Z,D}, μ=[$_1$]$_1$,M_1=Z,



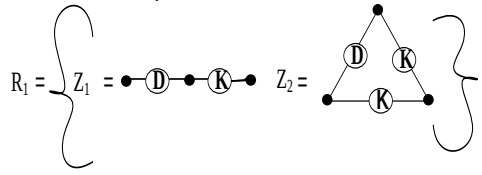
P_1=(true,out),F_1= (D,notout),(K,notout)

The corresponding language L(Π) is the set of all paths P_n and cycles C_n with n≥3 and their generation is same as that of hyperedge replacement graph grammar in Fig 2. Z has two rules but one of them is applied to get cycles or paths.

4.2 Generation of Paths and Cycles in Double membrane.

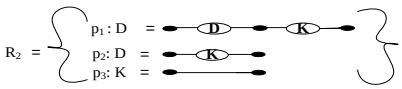
Π=(V,T,μ,M_1, ,(R_1,P_1,F_1), (R_2,P_2,F_2) (2, 2))

V={Z_1,Z_2,D}, μ=[$_1$[$_2$]$_2$]$_1$,M_1=Z_1,Z_2



P_1=(true,out),(true,in) F_1= (D,notout),(K,notout)

© 2019, IJCSE All Rights Reserved



P_2=(true,out),F_2= (D,notout),(K,notout)
The corresponding language L(Π) is the set of all paths P_n and cycles C_n with n≥3.Here the rule R_1 has two non terminals Z_1 and Z_2 that can be applied simultaneously (maximal parallelity) and enter the inner membrane to get paths and cycles. Here the generative power is more when compared with the single membrane P system

V. LEARNING ALGORITHM FOR GENERATING PATHS AND CYCLES

The grammatical inference algorithm in [8], based on hyperedge replacement to infer cycles results in many decompositions which makes the algorithm highly non-deterministic,For special classes of graphs namely paths and cycles , an alternative method is given to infer the exact grammar using edge contraction between adjacent vertices.

5.1. Algorithm:

INPUT:A positive presentation of Cycles and Paths (P_n and C_n n≥3) are given as inputs.

OUTPUT:A Sequence of Production rules that generate input samples and the classes of Paths and Cycles.
PROCEDURE:
Let G_i be the Input Samples.
Let Grammar,**Gr** = {{S}, P_2, (S, G_i) / i=1 to n, (S, 0)˙}
Initialize **Newprod**=ϕ
Initialize **RENAME NT**={S}
Initialize **REDUCE**= ϕ
Let i=1
Begin
For each G_i **do**
Begin
If G_i =C_3 or P_3
then DECOMPOSE (G_i)
else
Apply edge contraction between the adjacent vertices until C_3 or P_3 is obtained then **DECOMPOSE (G_i)**
End
Begin DECOMPOSE (G_i)
Introduce a new non-terminal D^j as a hyperedge of type 2, not occurring before, where j is a natural number such that
(S, G_i^1) ∪ (D^j, G_i^2) = (S, G_i) where k=1,2…
DECOMPOSE (G_i) = (S, G_i^1) ∪ (D^j, G_i^2) -(S,G_i).
Newprod = Newprod ∪ DECOMPOSE (Gi)
Again, Decompose G_i^2 such way that
(D^j, G_i^3) ∪ (D^{j+1}, G_i^4) and G_i^4 is path of length two.
DECOMPOSE (G_i^2) = (D^j, G_i^3) ∪ (D^{j+1}, G_i^4) -(D^j, G_i^2).
Newprod = Newprod ∪ DECOMPOSE (G_i^2)
End DECOMPOSE (G_i)
Gr = Gr ∪ Newprod
End
RENAME NT (D^j)
= {S=S, D^1=D^2=D^3=…=D}.
This function is similar to RENAME operation of Jelstch and Kreowski [4].
Gr=Gr ∪ Newprod ∪ RENAME NT
REDUCE is where Repeated and redundant productions are identified and removed [8]
Gr = Gr **- REDUCE**
Gr= {S, D, P_2 NewProd, S} is the grammar obtained from the above algorithm.

5.2. Learning Cycles and Paths using the above algorithm:

Consider the input graphs {(S, C_4), (S, P_5), (S, P_4)}
Gr ={{S}, P_2, (S, G_i) / i=1 to 3, (S,0)˙}
Initialize Newprod=ϕ
Initialize RENAME NT={S}
Initialize REDUCE= ϕ
Where G_1= C_4, G_2= P_5, G_3= P_4.
The graph C_4 is not equal to C_3 or P_3. Edge Contraction Between adjacent vertices are applied to get C_3. Introduce a new non-terminal D^1 as a hyperedge of type 2. Again introduce a new non-terminal D^2 as a hyperedge of type 2 such that G_1^4 is a path of length two such that (S, G_1^1) ∪(D^1, G_1^2)
DECOMPOSE (G_1) = (S, G_1^1) ∪(D^1, G_1^2)- (S, G_1).
Newprod = Newprod ∪ DECOMPOSE (G_1)
And again, Decompose G_i^2 such that
(D^1, G_1^3) ∪(D^2, G_1^4).
DECOMPOSE (G_1^2) = (D^1, G_1^3) ∪ (D^2, G_1^4) – (D^1, $G_1^{2)}$.
Newprod = Newprod ∪ DECOMPOSE(G_1^2)
DECOMPOSE(G_1) is given in Figure 8.

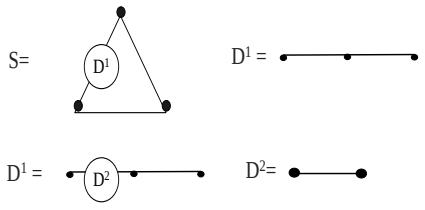


Figure 8

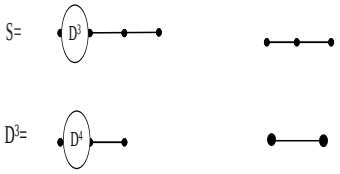


Figure 9

© 2019, IJCSE All Rights Reserved

Next input is G_2, applying the above process Figure 9 is obtained. $(S, G_2^1) \cup (D^3, G_2^2) = (S, G_2)$,
$DECOMPOSE(G_2) = (S, G_2^1) \cup (D^3, G_2^2) - (S, G_2)$.
Newprod = Newprod ∪ DECOMPOSE (G_2) and Again $(D^3, G_2^3) \cup (D^4, G_2^4)$.
$DECOMPOSE (G_2^2) = (D^3, G_2^3) \cup (D^4, G_2^4) - (D^3, G_2^2)$..
Newprod = Newprod ∪ DECOMPOSE (G_2^2)

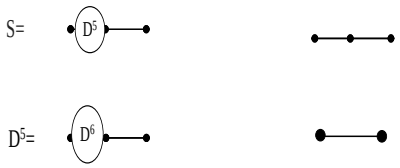


Figure 10

For G_3, applying the above process Figure 10 is obtained. $(S, G_3^1) \cup (D^5, G_3^2) = (S, G_3)$ and $DECOMPOSE (G_3) = (S, G_3^1) \cup (D^5, G_3^2) - (S, G_3)$.
Newprod= Newprod ∪ DECOMPOSE(G_3) and Again $(D^5, G_3^3) \cup (D^6, G_3^4)$.
$DECOMPOSE (G_3^2) = (D^5, G_3^2) \cup (D^6, G_3^4) - (D^5, G_3^2)$.
Newprod= Newprod∪DECOMPOSE (G_3^2)
Then RENAME NT $(D^j) = \{S=S, D^1=D^2=D^3=D^4=D^5=D^6=D\}$ is given in Figure 11, 12 and 13.

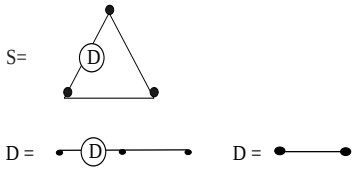


Figure 11 RENAME NT of C_4

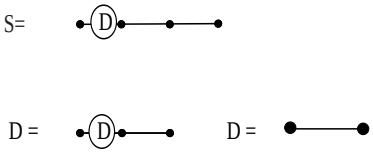


Figure 12 RENAME NT of P_5

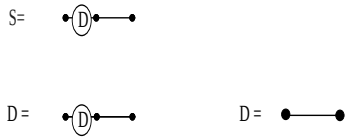


Figure 13 RENAME NT of P_4

Gr=Gr ∪ Newprod ∪ RENAME NT. Repeated and Redundant productions are identified and removed in REDUCE. The rules corresponding to P_5 can reduced when renaming is given to the non-terminals. Hence removing the REDUCE productions derives the sample inputs and the class of all Cycles and Paths. Gr = Gr-REDUCE.

Applying the above algorithm, the required production rules are obtained in Figure 14.

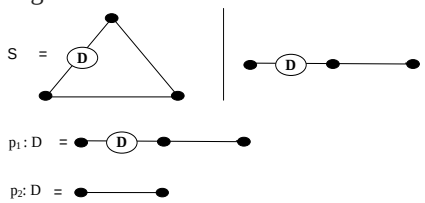


Figure 14

5.3 Correctness of the Algorithm [8]:

1. Each sample can be derived from the axiom of an inferred grammar.
2. Each production either being an initial one or one obtained by decompositions and renaming can be used for deriving one of the samples.
3. The axiom of an inferred grammar is $(S, 0)^{\bullet}$ or some renaming of it because the axiom has this form initially and the RENAME operation is the only one affecting the axiom.

A grammar with the above properties is called samples composing. Hence our grammar is samples composing as it satisfies the above conditions. Since the HRG of cycles and paths is a subclass of the class of hyperedge replacement grammars in [4] it is decidable, that cycles and paths can be inferred.

VI. CONCLUSION

An grammatical inference algorithm for inferring Paths and Cycles using Hyperedge Replacement Graph grammars is given which uses edge contraction between the adjacent vertices, followed by the four operations from which the required grammar is obtained. Our future work is to extend it further for varied classes of graphs and also for different graph operations namely corona product of graph, graph join and composition of graphs.

REFERENCES

[1] A.Habel , H.J. Kreowski ..:"*May We introduce to you: Hyperedge Replacement".* Lecture Notes in Computer Science, vol.**291**,pp **15-26** ,**1987**.

[2] G. Rozenberg.: "*Handbook of graph grammars and computing by graph transformation"*, vol **I** World Scientific ,**1997**.

[3] J,Engelfriet, " *Context- free graph grammars"*. In: G.Rozenberg , A.Salomaa, (eds) "*Handbook of Formal Languages*", Computer Science.Springer **4, 18 11**, **2006**.

[4] G.Paun*" A guide to membrane computing"*, Theoretical Computer Science, Vol.**287, 73-100, 2002**,.

[5] Colin de la Higuera, "*Current Trends in Grammatical Inference"*, Lecture Notes in Computer Science, 1876,**28-31, 2000**.

[6] G.Paun, Rozenberg.G.,Saloma .A., "*The oxford Handbook of Membrane Computing*",**2010**

© 2019, IJCSE All Rights Reserved

[7] Meena Parvathy Sankar, N.G.David ,D.G.Thomas *,"Hyperedge replacement Graph P system",* Proceeding BIC-TA'11, Proceedings of the 2011 Sixth International Conference on Bio-Inspired Computing: theories and Applications,**2011.**

[8] E.Jeltsch , H.J Kreowski.: "*Grammatical Inference based on Hyperedge Replacement"* ,Lecture Notes in Computer Science, vol **32**, pp **461-474**, **1990**.

Authors Profile

Mrs.Thanga Murugeshwari .V received the M.Sc. Degree in Mathematics in the year 2011, M.Phil. Degree in 2012. She is currently pursuing her Ph.D in Queen Mary's College,Chennai-under University of Madras. Her research area includes Formal Languages and Learning theory.

Dr.Mrs. Emerald Princess Sheela J.D. received the M.Sc. Degree in Mathematics in the year 1991, M.Phil. Degree in 1993 and the Ph.D. Degree in 2001 from the University of Madras. She has been on the teaching faculty as Assistant Professor, Department of Mathematics, since 2001 in various colleges and is currently working in Queen Mary's College, Chennai -4. She ha s presented her research papers in two International Conferences in USA, one in Malaysia and one in UK. She is the author or co-author of about 20 research papers in her field of interest which includes formal languages and automata theory, cryptography and learning theory. She is also guiding three research students for the award of Ph.D. Degree in the University of Madras.

© 2019, IJCSE All Rights Reserved