GRAPH OL-SYSTEMS AND RECURRENCE SYSTEMS ON GRAPHS*

K. Culik and A. Lindenmayer**
Department of Computer Science
University of Waterloo, Waterloo, Ontario, Canada
and
Theoretical Biology Group
University of Utrecht, Utrecht, The Netherlands

CS-74-18

October 1974

This is an extended abstract of a paper to be presented at the Eighths Hawaii Conference on Systems Science, January, 1975.

* This work has been supported by the National Research Council of Canada, Grant No.A7403

** Visiting Professor at the University of Waterloo in the fall term 1974.

GRAPH OL-SYSTEMS AND RECURRENCE SYSTEMS ON GRAPHS*

K. Culik II and A. Lindenmayer**
Department of Computer Science
University of Waterloo, Waterloo, Ontario, Canada
and
Theoretical Biology Group
University of Utrecht, Utrecht, The Netherlands

Parallel rewriting systems on strings have in recent years been employed to model the development of multicellular filamentous organisms [1,2]. In the present paper we propose to generalize parallel rewriting to graphs in order to enable us to model the development of multidimensional organisms.

For biological reasons we take cells to be the basic units since they are known to be metabolically and genetically autonomous functional units of most organisms. The most relevant aspect of their autonomy is the fact that all cells of a given organism contain the same complement of DNA, which each cell receives from its mother cell and passes on to its daughter cells. Our systems incorporate this uniformly programmed and highly redundant aspect of multicellular development by applying the same rewriting rules to all cells in a given multidimensional array.

A cellular array is essentially a subdivision of a limited space into units which completely fill the space. Tessellations and maps are examples of two-dimensional cellular arrays. We wish to concern ourselves primarily with neighborhood aspects of such arrays, i.e. topological aspects, and not with their detailed metric descriptions. The timing and orientation of a cell division, for instance, is to be affected by the state of the cell and the states of neighboring cells, but not by their exact shapes, sizes and positions, as would be the case in a metric description. Biochemical and cell-physiological mechanisms make topological descriptions of cellular development the more plausible ones. This aspect of our models is expressed by our choice of graph representation of cellular arrays.

We choose a graph to be defined as a set of ordered pairs (directed edges) over a set of nodes. Both nodes and edges are to be labeled. This definition of graphs precludes multiple edges with the same label between the same ordered pair of nodes. Thus cases where cells in an array touch each other along more than one discontinuous boundary cannot be expressed by our graph notation. Such cases occur rather rarely in organisms, and we gain considerable clarity by omitting them.

The node labels in our graphs correspond to states of cells. The cells are considered to be finite automata, each cell changing its state at discrete time intervals according to its previous state and its inputs. The states and inputs of cells are interpreted as combinations of chemical and physical factors present in the cells or entering them.

In order to model development, new cells must be added to or taken away from arrays at certain times and places. Thus we need to allow the substitution of a graph for each node of the previous graph. These substitution (or rewriting) rules are considered to be extensions of the state transition functions of finite automata.

Purely topological descriptions of cellular arrays would not require edge labeling. However, sometimes we wish to include in our descriptions geometric properties other than only the neighborhood relation. Edge labeling and direction of edges enable us to express some simple geometric properties, for example, by choosing one edge label for horizontally touching neighbors, and another one for vertically touching ones.

By simultaneous application of node rewriting rules (productions) to all the nodes of a graph, and then by application of suitable connection rules, we obtain a new graph. By repetition of this procedure a set of graphs is generated, which constitutes a developmental graph language.

Mayoh [3,4] has proposed and demonstrated on some particular examples graph generating systems with connection rules based on node labels. We are now presenting a powerful and general mechanism for defining connection rules, dependent both on node labels and on the structure of the neighborhoods (e.g. ordering of neighbors).

Parallel rewriting systems on one-dimensional cellular arrays (represented by strings of symbols) have been called "OL-systems" if no interaction takes place among the cells, and "IL-systems" if there is interaction. Deterministic string generating L-systems are those which have a single production for each symbol, and propagating L-systems are those which do not allow erasing of symbols (no cell death).

* This work has been supported by the National Research Council of Canada, Grant No.A7403.

** Visiting Professor at the University of Waterloo in the fall term 1974.

Generating systems (grammars) for graphs or for multidimensional arrays (webs, cellular automata) have been proposed before (references would be too numerous to be included here), but all of these constructs were such that the graphs or arrays were either allowed to grow only at the edges or surfaces, or substitutions for nodes (subgraphs) were allowed only sequentially. For biological reasons we insist on simultaneous rewriting and on being able to add new cells in the interior of the array. We also wish to allow connections only between nodes which are either daughter nodes of the same mother node, or the mothers of which were connected in the previous graph.

Definition of propagating graph OL-systems (PGOL-systems)

For lack of space, instead of formal definitions (to be found in [5]), we present the needed notions here only informally.

As stated before, organisms in our systems are represented by directed graphs with labeled nodes and labeled edges, having a special node (the environmental node) labeled by e. Edges going to or from the environmental node shall be called outside edges and the others inside edges. Since we are not interested in naming individual nodes, we shall only be concerned with isomorphic classes of such graphs (in the following called abstract e-graphs).

An abstract e-graph over (Σ,Δ,Δ') is an isomorphic class of graphs with environmental nodes labeled e ($e \notin \Sigma$), all the other nodes labeled by elements of Σ, inside edges labeled by elements of Δ and outside edges by elements of Δ'. The family of all abstract e-graphs over (Σ,Δ,Δ') is denoted by $(\Sigma,\Delta,\Delta')_*$. The empty abstract e-graph (the abstract e-graph with single node labeled by e and no edges) is denoted by λ, and the set $(\Sigma,\Delta,\Delta')_* - \{\lambda\}$ is denoted by $(\Sigma,\Delta,\Delta')_+$. Whenever $\Delta = \Delta'$ we will write (Σ,Δ,Δ') as (Σ,Δ).

In our diagrams of abstract e-graphs the outside edges will be shown as free arrows and the environmental node will not be shown. The outside edges will in this context be called "hands".

We want to define now the coalescence of two abstract e-graphs into one by adding new edges. This will be done with the help of "stencils". A stencil is an abstract e-graph with a bipartition of its nodes (strictly speaking, with a bipartition of the nodes of any of the representants in this equivalence class). In consequence, each stencil has two subsets of nodes, and we shall call these the "source" and "target" nodes. The set of all stencils over (Σ,Δ,Δ') is denoted by $(\Sigma,\Delta,\Delta')^{\$}_*$.

An ordered pair of abstract e-graphs (or simply graphs) α, β can be joined (coalesced) according to the stencil γ into a new graph δ, denoted as $\alpha \xrightarrow{\gamma} \beta$, in the following way.

Only those pairs of nodes from α and β may be joined which have matching hands (i.e., one node in α has an outgoing and the other in β an incoming hand, or vice versa, with the same label). Which of such pairs of nodes are connected is determined by the stencil γ in the sense that after all new connections are made, γ is a subgraph of δ so that (1) the source nodes of γ are all in α, (2) the target nodes of γ are all in β, and (3) all the new edges in δ (between nodes in α and in β) are in γ. In the following, we shall call stencil γ applicable to the ordered pair α, β if α and β can be joined according to γ.

For a stencil γ let γ_S, γ_T denote the subgraphs induced by its source and target nodes, respectively. Let C be a set of stencils, and γ be in C. We say that γ is C-maximal for α, β if γ is applicable to α, β and if there is no η in C such that η is applicable to α and β, γ_S is a subgraph of η_S, and γ_T is a subgraph η_T.

Next we extend our definition of joining of graphs by a stencil to the case where two graphs α and β are joined with respect to a set of stencils C, giving rise to a set of graphs $D = \alpha \xrightarrow{C} \beta$, as follows:

$$D = \{\alpha \xrightarrow{\gamma} \beta : \gamma \in C \text{ and } \gamma \text{ is C-maximal for } \alpha,\beta\}$$

For the formal definition of both graph production systems and graph recurrence systems we need the notion of "graph expression". A <u>graph expression</u> over (Σ,Δ) denotes a set of graphs over (Σ,Δ) and it has actually the form of an abstract e-graph of which the nodes are labeled by graphs over (Σ,Δ), the inside edges are labeled by sets of stencils over (Σ,Δ), and the hands are labeled by elements of Δ. In other words, a graph expression over (Σ,Δ) is an abstract e-graph over (Ω,π,Δ) where Ω is a finite subset of $(\Sigma,\Delta)_*$, and π is a finite set of subsets of $(\Sigma,\Delta)^{\$}_*$.

A graph expression A over (Σ,Δ) defines the set of graphs over (Σ,Δ) denoted by D(A). The definition is as follows:

Let the nodes of A except the environmental node be indexed by numbers $1,\ldots,n$, and let graph α_i be the label of the i-th node ($\alpha_i \in \Omega$) for $1 \le i \le n$. Then δ is an element of D(A) if there exists a partition of the nodes of δ into n+1 sets, namely $\{e\}, V_1,\ldots,V_n$, so that

1) The subgraph of δ induced by V_i is the graph α_i, for each $1 \le i \le n$.

2) For every edge of A the following must hold: let the edge go from node indexed by i to node indexed by j and let it be labeled by stencil-set H. There must exist an element γ of the set of graphs $\alpha_i \xrightarrow{H} \alpha_j$ such that the subgraph of γ induced by the non-environmental nodes of γ must be identical to the subgraph of δ induced by $V_i \cup V_j$.

3) There is an in(out)-going hand labeled h to (from) a node a of δ, say a is node of α_i, iff there is an in(out)-going hand with label h to (from) a in α_i, and to (from) the mother node indexed by i in A.

Having given an informal definition of graph expressions, we can now define a <u>propagating graph</u>

OL-system (PGOL-system). A PGOL-system G is a quintuple $<\Sigma,\Delta,P,C,S>$, where

Σ is an alphabet of node labels;
Δ is an alphabet of edge labels;
P is a finite subset of $\Sigma \times (\Sigma,\Delta)_+$ of productions;
C is a finite subset of $\Delta \times (\Sigma,\Delta)^s_+$ of connection rules;
S in $(\Sigma,\Delta)_+$ is the axiom (initial graph).

Productions and connection rules are written in the form $a \mapsto \alpha$. The set P must be complete, i.e. for each a in Σ there is an α so that $a \mapsto \alpha$ is in P.

For γ,δ in $(\Sigma,\Delta)_+$ we write $\gamma \underset{G}{\Rightarrow} \delta$ if there exists a graph expression W such that:

i) W is obtained by relabeling of γ so that each occurrence of a node label from Σ-{e}, say a, is replaced by α for some $a \mapsto \alpha$ in P, and each edge label, say b, of an inside edge is replaced by the set of stencils $\{B:b \mapsto B$ is in C$\} \cup \{\lambda\}$.

ii) δ is in D(W).

The reflexive and transitive closure of relation $\underset{G}{\Rightarrow}$ is denoted by $\underset{G}{\Rightarrow}^*$.

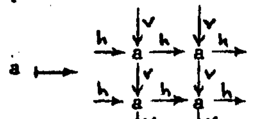
The graph language generated by G is denoted by L(G) and defined as $\{\alpha:S \underset{G}{\Rightarrow}^* \alpha\}$.

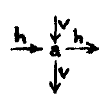
A PGOL-system (Σ,Δ,P,C,S) is called deterministic if P and C satisfy the following conditions:

i) For every a in Σ there is exactly one α so that $a \mapsto \alpha$ is in P.
ii) Let an edge labeled by h and pointing from a node labeled a_1 to a node labeled a_2 occur in S, P or C; let $a_1 \mapsto \alpha_1$ and $a_2 \mapsto \alpha_2$ be in P; and let $H = \{B:h \mapsto B \in C\}$. There must be at most one γ in H such that γ is H-maximal for α_1, α_2.

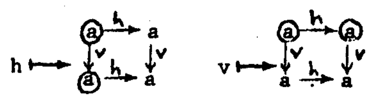
In the following example we represent abstract e-graphs by diagrams showing only node labels, not the nodes themselves. In stencils the source node labels are circled.

Example. Let $G = <\{a\},\{h,v\},P,C,S>$ be a PGOL-system, where set P consists of the single production: S is the graph:

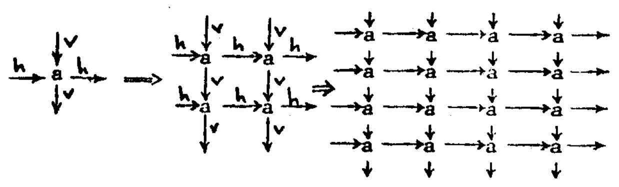




and C is the set of the following connection rules:



This, clearly, is a deterministic PGOL-system. The first two derivation-steps are shown below, with edge labels omitted in the last diagram:



If we interpret graphs in L(G) as planary maps with every a represented by a square of equal size, and h and v by horizontal and vertical relative positions of neighbors, then we seem to generate by this system square grids of 4^n units. In fact, the reader can verify that the stencils on the right-hand sides of connection rules give us exactly those graphs which correspond to square grids of 4^n units, for all $n \geq 0$. Biological applications of graph OL-systems are also available (see [6]), but cannot be presented in this abstract.

Results on PGOL-systems

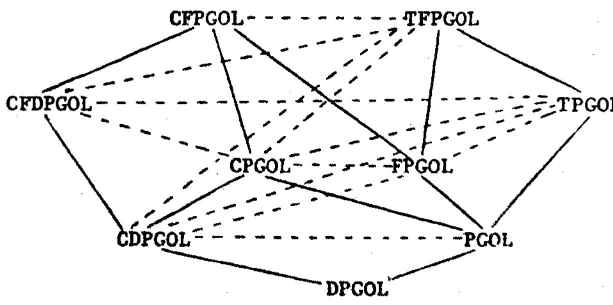
Various special types or modifications of string OL-systems have been studied, see e.g. [7] or [8] to which we refer the reader for formal definitions of "operators": F(finite number of axioms), D(deterministic), T(table) and C(codings or literal homomorphisms). We want to consider these "operators" and their combinations also for PGOL systems. D has already been defined, the meaning of F and T is obvious, for C we need to say, informally, that we will consider here only codings of node labels and not of edge labels.

We shall use the same notation for the families of graph languages generated by various types of systems as is common for the corresponding families of languages (see e.g. [7]) except that the names will have suffix GOL rather than OL. For example, the family of graph languages generated by deterministic table propagating GOL-systems with finite number of axioms will be denoted by DTFPGOL.

Theorem 1 Let X and Y be any combinations of "operators" D, F, T and C. If the classes of string languages XPOL and YPOL are incomparable classes of graph languages, then the XPGOL and YPGOL are incomparable.

Theorem 2 If XPGOL $\subseteq$ YPGOL and XPOL $\subsetneq$ YPOL then XPGOL $\subsetneq$ YPGOL.

Idea of proof of Theorems 1 and 2: For every combination of operators X we define a special subclass of XPGOL systems such that any XPOL-system can be simulated by a special XPGOL-system, and vice versa.



In the above figure we have summarised the proper containment (solid lines - if A is below B then $A \subsetneq B$) and incomparability (dotted lines) results for classes of XPGOL languages. They follow from the above two theorems, since it can be shown that the same diagram (when omitting G) is valid for classes of string languages.

We call a PGOL-system bifurcating if the right-hand side of its every production has at most two nodes.

Theorem 3 For every PGOL-system $G = (\Sigma,\Delta,P,C,S)$ there exists a constant k and a bifurcating PGOL-system $G' = (\Sigma',\Delta,P',C',S)$ such that $S \overset{n}{\underset{G}{\Rightarrow}} W$ iff $S \overset{nk}{\underset{G'}{\Rightarrow}} W$, where $\overset{j}{\underset{G}{\Rightarrow}}$ means "derives in j steps".

Almost all living cellular developmental systems are bifurcating. However, the above theorem provides justification for studying and using non-bifurcating systems as well, since the developmental behavior of such systems corresponds to infrequent observations of the detailed sequence of the actual bifurcating behavior.

Recurrence systems on graphs

In this section we can give only a brief outline of the material that is contained in the full paper [5]. Using a generalisation of graph expressions and in addition similarly defined stencil expressions we can define graph recurrence systems. Analogously to string recurrence systems [9], we need one or more recurrence equations for defining sets of graphs. But in the case of graph recurrence systems we also need one or more recurrence equations for sets of stencils to be used as connection rules in each of the recurrence equations.

Example. Now, we give a recurrence system (with the distinguished variable A) for the graph language generated by the PGOL-system in the previous example. Note that $\bar{H}$ and $\bar{V}$ are constant stencils.

$$A_0 = \begin{tikzcd} & {} \arrow[d, "v"] & \\ {} \arrow[r, "h"] & a \arrow[r, "h"] \arrow[d, "v"] & {} \\ & {} & \end{tikzcd} \qquad A_{n+1} = \begin{tikzcd} & {} \arrow[d, "v"] & {} \arrow[d, "v"] & \\ {} \arrow[r, "h"] & A_n \arrow[r, "H_n"] \arrow[d, "V_n"] & A_n \arrow[r, "h"] \arrow[d, "V_n"] & {} \\ {} \arrow[r, "h"] & A_n \arrow[r, "H_n"'] \arrow[d, "v"] & A_n \arrow[r, "h"] \arrow[d, "v"] & {} \\ & {} & {} & \end{tikzcd}$$

$$H_0 = \begin{tikzcd} & {} \arrow[d, "v"] & {} \arrow[d, "v"] & \\ {} \arrow[r, "h"] & \textcircled{a} \arrow[r, "h"] \arrow[d, "v"] & a \arrow[r, "h"] \arrow[d, "v"] & {} \\ & {} & {} & \end{tikzcd} \qquad H_{n+1} = \begin{tikzcd} {} \arrow[d, "v"] \\ H_n \arrow[d, "\bar{V}"] \\ H_n \arrow[d, "v"] \\ {} \end{tikzcd}$$

$$V_0 = \begin{tikzcd} & {} \arrow[d, "v"] & \\ {} \arrow[r, "h"] & \textcircled{a} \arrow[r, "h"] \arrow[d, "v"] & {} \\ {} \arrow[r, "h"] & a \arrow[r, "h"] \arrow[d, "v"] & {} \\ & {} & \end{tikzcd} \qquad V_{n+1} = \begin{tikzcd} {} \arrow[r, "h"] & V_n \arrow[r, "\bar{H}"] & V_n \arrow[r, "h"] & {} \end{tikzcd}$$

$$\bar{H} = \begin{tikzcd} {} \arrow[r, "h"] & \textcircled{a} \arrow[r, "h"] \arrow[d, "v"] & a \arrow[r, "h"] \arrow[d, "v"] & {} \\ {} \arrow[r, "h"] & \textcircled{a} \arrow[r, "h"] & a \arrow[r, "h"] & {} \end{tikzcd} \qquad \bar{V} = \begin{tikzcd} {} \arrow[d, "v"] & {} \arrow[d, "v"] \\ \textcircled{a} \arrow[r, "h"] \arrow[d, "v"] & \textcircled{a} \arrow[d, "v"] \\ a \arrow[r, "h"] \arrow[d, "v"] & a \arrow[d, "v"] \\ {} & {} \end{tikzcd}$$

In the following we consider only so-called λ-free recurrence systems on graphs. In these systems the empty graph λ can only be used as a constant stencil but not as an initial value nor at the right side of any equation.

The following theorem can be proved [5] which is a generalisation of a similar result for strings.

Theorem 4 Every PGOL-language can be defined by a recurrence system.

For strings there is a complementary result, namely, that every language L described by a recurrence system can be expressed as $L_1 \cap \Sigma_1^*$ where L_1 is an OL-language over Σ and $\Sigma_1 \subseteq \Sigma$. This result cannot be extended to graphs. We see from the following decidability results that recurrence systems for graphs are much more complex than PGOL-systems. So the generalisation of recurrence systems from strings to graphs is much "stronger" than the generalisation of production systems from strings to graphs.

Theorem 5 Given a PGOL-system (FTPGOL-system) G and an abstract graph α it is decidable whether α is a subgraph of some element of L(G).

Theorem 6 Given a recurrence system and an abstract graph α it is recursively undecidable whether α is a subgraph of some element of the language defined by the given recurrence system.

Theorems 5 and 6 hold also when we ask whether α is a full subgraph rather than subgraph. A full subgraph of β is a subgraph of β induced by a subset of nodes of β. Since neither PGOL systems nor λ-free recurrence systems allow erasing we have the following.

Theorem 7 The membership problem is decidable for both PGOL-systems and recurrence systems on graphs.

In [5] a definition is given for propagating graph systems with interactions (PGIL-systems) in which the left-hand side of every production is a graph with a distinguished node. The distinguished node is the one to be replaced and the graph provides the restrictional context for the replacement.

REFERENCES

1. A. Salomaa, Formal languages, Part 2, Section 13, Academic Press, New York, 1973.

2. G.T. Herman and G. Rozenberg, with a contribution by A. Lindenmayer, Developmental Systems and languages, North-Holland Publ.Co., Amsterdam (in press).

3. B.H. Mayoh, Mathematical model for cellular organisms, Dept. of Comp. Sci., Aarhus Univ., Denmark, Rep. No. DAIMI PB-12, Apr.1973, 38 pp.

4. B.H. Mayoh, "Multidimensional Lindenmayer organisms", in L Systems, edited by G. Rozenberg and A. Salomaa, Lect. Notes in Computer Science No.15, Springer Verlag, Heidelberg (in press).

5. K. Culik II, and A. Lindenmayer, "Parallel rewriting systems for graphs", (paper in preparation).

6. A. Lindenmayer, and K. Culik II, "Graph systems and languages for multidimensional cellular development" (paper in preparation).

7. M. Nielsen, G. Rozenberg, A. Salomaa, and S. Skyum, Nonterminals, homomorphisms and codings in different variations of OL systems, Parts I and II", Dept. of Comp. Sci., Aarhus Univ., Denmark, Rep. No. DAIMI PB-21, Jan. 1974, 50 pp.

8. K. Culik II, and J. Opatrny, "Literal homomorphisms of OL-languages", Int. J. Computer Math. (in press), extended abstract in Proceedings of the 1974 conference on biologically motivated automata theory, pp.50-53.

9. G.T. Herman, A. Lindenmayer, and G. Rozenberg, "Descriptions of developmental languages using recurrence systems", Math. Systems Theory (in press).