PARALLEL REWRITING ON GRAPHS AND MULTI-DIMENSIONAL DEVELOPMENT*

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

CS-74-22

November 1974

* This work has been supported by the National Research Council of Canada, Grant No. A7403. A preliminary version of this paper has been presented to the 8th Hawaii International Conference on Systems Science, January 1975.

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

PARALLEL REWRITING ON GRAPHS AND MULTIDIMENSIONAL DEVELOPMENT*

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

Abstract. As an extension of parallel rewriting systems on strings of symbols (L-systems), graph generating systems (graph L-systems) and graph recurrence systems are defined, by which the development of multidimensional organisms can be modelled. Organisms are represented by directed graphs with labeled nodes and labeled edges. The nodes stand for cells, their labels for cellular states; the edges for connections between neighbors, their labels and directions for geometric relationships. In generating systems, a new graph is produced by simultaneous substitution of graphs for nodes according to a finite set of productions, and by connecting their nodes by adding new edges according to a finite set of connection rules (using "stencils"). In graph recurrence systems larger and larger graphs are defined by recursive formulas on graphs and stencils. Results are given on the hierarchy of graph language families, as well as on some decidability problems concerning them.

---

* This work has been supported by the National Research Council of Canada, Grant No. A7403. A preliminary version of this paper has been presented to the 8th Hawaii International Conference on Systems Science, January 1975.

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

1. Introduction

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.

We also rule out cases where a cell touches itself (having an edge from a node to itself in the graph) for the same reasons.

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.

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 on node and edge labels, edge directions, and on the structure of the neighborhoods.

Edge labels and edge directions play a generative role in our systems just as node labels do. While the node labels control which subgraph is substituted for a particular node, the edge directions and labels control how pairs of substituted subgraphs get connected with each other. From a biological point of view the directedness of an edge corresponds to polarity in the connection between two neighbor cells. Lack of polarity can be expressed by having two edges in opposite directions between nodes. Edge labels of more than one kind in a graph generating system imply biologically that cells have mechanisms by which they can distinguish between various kinds of connections among neighbors. For instance, if it can be assumed that in a certain animal dorso-ventral connections between pairs of cells are distinguished from bilateral distal-proximal connections, and from head-tail axis connections, then we are justified to model the development of this animal by a graph generating system having three edge labels. In other words, there has to be a cellular or molecular mechanism by which different kinds of neighborhoods are determined.

Another way of looking at edge labels and directions is that they enable us to include additional geometric properties in our graphs other than topological neighborhoods. For example, by choosing one edge label for horizontally touching neighbors, and another for vertically touching ones, we can obtain an obvious representation of some sets of planar graphs, namely, representations on rectangular grids (see Example 1).

We wish to allow connections to be generated only between nodes which are either within the subgraph substituted for a node, or between nodes of which the "mother" nodes were connected in the previous step.

Generating systems (grammars) for graphs or for multidimensional arrays (webs, cellular automata) have been proposed before e.g. [5 - 11], 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.

After completing the draft of this paper, we have seen a recent work [12] in which two-dimensional cellular generating systems are defined in such a way that they fulfill the two biologically motivated conditions mentioned above. In these systems, the cells divide simultaneously into at most two new cells, and new structures are defined by orienting the newly formed boundaries according to the configuration of all of the neighboring cells. Cells can touch each other along more than one boundary, and they can also touch themselves. In order to have finite number of rules, it is required in these systems that before each computation the number of neighbors of each cell is counted, and if it exceeds certain bound, all subsequent divisions of that cell and the cells in its neighborhood must be oriented in such a way that the number of its neighbors does not increase. The requirement to count the number of neighbors of each cell is a rather artificial one from a biological point of view, although it certainly is physically impossible for an actual cell to have unboundedly many neighbors and there must be a mechanism which limits their number. In our systems no bounds are imposed on numbers of neighbors, except by the productions and connection rules chosen. The reason why we can still define our systems

by finite numbers of connection rules is that we connect the substituted subgraphs pairwise with each other, rather than considering all of the neighboring subgraphs in the connection rules. However, our recurrence systems also allow the latter kinds of connection rules, and in this case infinite sets of such rules are defined by recurrence formulas. In contrast to the systems defined in [12], our graph systems are not limited to two-dimensional structures.

2. Preliminaries

Let Σ be an alphabet (a finite, nonempty set of symbols). An index set $X = \{X_a\}_{a\in\Sigma}$ is a family of sets. For each $a \in \Sigma$, X_a is a set. We write $X_\Sigma = \bigcup_{a\in\Sigma} X_a$. We also write $Z = X \cup Y$, or in more detail $\{Z_a\}_{a\in\Sigma} = \{X_a\}_{a\in\Sigma} \cup \{Y_a\}_{a\in\Sigma}$, iff $Z_a = X_a \cup Y_a$ for each $a \in \Sigma$. Similarly for other Boolean operations and relations, e.g. inclusion.

A concrete (directed) graph over Σ, Δ is a pair $(\{V_a\}_{a\in\Sigma},\{E_b\}_{b\in\Delta})$ in short (V,E) where

(i) $V_a \cap V_{a'} = \phi$ for $a \neq a'$

(ii) $E_b \subseteq V_\Sigma^2 - I_\Sigma$ for each $b \in \Delta$, where $I_\Sigma = \{(x,x):x \in V_\Sigma\}$.

Thus our notation has the following meaning:

V_Σ is the set of (all) nodes;

E_Σ is the set of (all) edges;

V_a is the set of nodes labeled by a, for every $a \in \Sigma$;

E_b is the set of edges labeled by b, for every $b \in \Delta$.

As noted before, V and E are index sets of nodes and edges, respectively. The index sets of nodes and edges of a concrete graph α over Σ, Δ will always be denoted by V^α and E^α, respectively.

Since we are not interested in naming individual nodes, we will be mainly concerned with isomorphic classes of concrete graphs. An isomorphic class of concrete graphs will be called an abstract graph or in short graph.

The class of all concrete graphs isomorphic to a concrete graph α is denoted by $[\alpha]$. Concrete graph α is called a representant of the abstract graph $[\alpha]$. The empty abstract graph is the graph with no nodes or edges and it is denoted by λ.

A concrete graph α is a subgraph of a concrete graph β, written $\alpha \leq \beta$, if $V^\alpha \subseteq V^\beta$ and $E^\alpha \subseteq E^\beta$. For proper subgraph we write $\alpha < \beta$.

An abstract graph A is a subgraph of an abstract graph B if there exist concrete graphs α, β so that $A = [\alpha]$, $B = [\beta]$, and $\alpha \leq \beta$.

The subgraph of a concrete graph α induced by a subset of its nodes X ($X \subseteq V^\alpha_\Sigma$) is denoted by $\langle X \rangle_\alpha$, and is defined as the concrete graph (V,E) where $V_a = V^\alpha_a \cap X$ for each $a \in \Sigma$ and $E_b = \Sigma^\alpha_b \cap X^2$ for each $b \in \Delta$.

For concrete graphs α, β, we say that α is a full subgraph of β if $V^\alpha \subseteq V^\beta$ and $\langle V^\alpha \rangle_\beta = \alpha$. For abstract graphs A, B we say that A is a full subgraph of B if there exist representants α, β of A, B respectively so that α is a full subgraph of β.

For concrete graphs α, β, we write $\alpha \cup \beta = (V^\alpha \cup V^\beta, E^\alpha \cup E^\beta)$.

3. Definition of graph generating systems

3a. Definition of e-graphs and e-stencils

Organisms in our systems are represented by concrete graphs with labeled nodes and labeled edges, having a special node (the environmental node) labeled by e. Let Σ, Δ, Δ' be alphabets and $e \notin \Sigma \cup \Delta \cup \Delta'$.

A concrete e-graph over Σ, Δ, Δ' is a concrete graph (V,E) over $\Sigma \cup \{e\}$, $\Delta \cup \Delta'$, where $V_{\Sigma \cup \{e\}}$ contains symbol ∞ (called the environmental node), $V_e = \{\infty\}$, $E_b \subseteq (V_\Sigma)^2$ for each $b \in \Delta$, and $E_b \subseteq (V_\Sigma \times \{\infty\}) \cup (\{\infty\} \times V_\Sigma)$ for each $b \in \Delta'$. Accordingly, ∞ is the only node labeled by e, all the edges not incident with ∞ (called in the following inside edges) are labeled by symbols of Δ, and all the edges incident with ∞ (called outside edges) are labeled by symbols of Δ'. The family of all concrete e-graphs over Σ, Δ, Δ' is denoted by $(\Sigma,\Delta,\Delta')_*$ or in short $(\Sigma,\Delta)_*$ if $\Delta = \Delta'$.

If α is a concrete e-graph, then $[\alpha]$ is an abstract e-graph (or in short e-graph).

The family of all abstract e-graphs over Σ,Δ,Δ' is denoted by $[\Sigma,\Delta,\Delta']_*$ or in short $[\Sigma,\Delta]$ if $\Delta = \Delta'$. We write $[\Sigma,\Delta,\Delta']_+ = [\Sigma,\Delta,\Delta']_* - \{\lambda\}$.

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".

In the next section we define the joining of two concrete (abstract) e-graphs into one by adding new edges. This will be done with the help of "stencils". Intuitively, a concrete stencil is a concrete graph with bipartition of nodes into "source" and "target" nodes. Since we want to define also abstract stencils we will give this partition by using an extended alphabet of node labels. We will add symbols s and t to the labels of source and target nodes, respectively.

For an alphabet Σ, we define $\Sigma_s = \{(a,s):a \in \Sigma\}$, $\Sigma_t = \{(a,t):a \in \Sigma\}$ $\Sigma_{st} = \Sigma_s \cup \Sigma_t = \Sigma \times \{s,t\}$.

A concrete e-stencil over Σ,Δ,Δ' is a concrete e-graph over $\Sigma_{st},\Delta,\Delta'$. Thus the family of all e-stencils over Σ,Δ,Δ' is the set $(\Sigma_{st},\Delta,\Delta')_*$. Let $\alpha = (V^\alpha,E^\alpha)$ be a stencil over Σ,Δ,Δ'. Then the set of nodes $V^\alpha_{\Sigma_{st}}$ of α is by definition bipartitioned into the source set $V^\alpha_{\Sigma_s}$ and the target set $V^\alpha_{\Sigma_t}$.

An abstract e-stencil (in short stencil) over Σ,Δ,Δ' is an abstract e-graph over $\Sigma_{st},\Delta,\Delta'$. Thus the set of all abstract stencils over Σ,Δ,Δ' is the set $[\Sigma_{st},\Delta,\Delta']_*$.

The concrete e-graph β obtained from the concrete e-graph $\alpha = (V^\alpha,E^\alpha)$ over Σ,Δ by merging a subset of its nodes X containing ∞ $(X \subseteq V^\alpha_\Sigma \cup \{\infty\})$ into ∞ is denoted by $mer(X,\alpha)$ and formally defined as the graph (V^β,E^β) where

(i) $V^\beta_a = V^\alpha_a - X$ for each a in Σ,

(ii) $E^\beta_b = (E_b \cap (V^\beta_\Sigma)^2) \cup \{(\infty,x):x \in V^\beta_\Sigma, y \in X,(y,x) \in E^\alpha_b\} \cup$

$\{(x,\infty):x \in V^\beta_\Sigma, y \in X,(x,y) \in E^\alpha_b\}$.

Note that no loops of length 1 are created by merging, we are not considering graphs with such loops.

We need a mapping g from concrete e-stencils over Σ,Δ,Δ' to concrete e-graphs over Σ,Δ,Δ' which discards the second components of node-labels. Formally, for α in $(\Sigma_{st},\Delta,\Delta')_*$, where $\alpha = (V,E)$, we define $g(\alpha) = (V',E)$ where $V'_a = V_{(a,s)} \cup V_{(a,t)}$ for each $a \in \Sigma$ and $V'_e = V_e = \{\infty\}$.

3b. Definition of joining pairs of concrete e-graphs

In a graph production system (defined in the next section), a new abstract e-graph is produced from a previous one by simultaneous substitutions of abstract e-graphs for each node in a previous abstract e-graph, after which step the substituted graphs must pairwise be connected with each other. We shall sometimes call the substituted abstract e-graphs "daughter graphs", and the nodes for which they are substituted the "mother nodes". The joining of an ordered pair of daughter graphs α,β takes place if and only if their mother nodes were connected by some edge labeled b, and directed from the mother of α to the mother of β, and there is an abstract e-stencil for b among the connection rules of the system which is applicable to the pair α,β.

The procedure of checking whether a given stencil γ is applicable to daughter graphs α and β can be imagined the following way. We draw the graphs α and β side-by-side on one sheet of transparent paper, and similarly the stencil γ on another sheet. We lay the first sheet over the second and ascertain first of all whether all the source nodes of γ are present in α, and all the target nodes of γ are present in β. Secondly, we make sure that all the edges which connect source nodes of γ are present between the corresponding nodes of α, and similarly that all the edges among the target nodes of γ are present in β. Finally, we observe for every edge labeled b in γ which connects some source node with some target node, or vice versa, whether there is a pair of "matching hands" with labels b on the corresponding nodes of α and β. By matching hands we mean two edges of the same label, one of which is directed to the environmental node and the other is directed away from it. This last requirement allows us to join up

only those nodes of daughter graphs which bear properly matched in- and out-going edges to and from the environment. Note that each substituted daughter graph is assumed to have its own environment, implied by it being an e-graph, in addition to the fact that the graph standing for the whole organism has its environment. One particular hand on a node in a daughter graph can be used in the joining of that node by several applicable stencils.

Once it has been established that a particular stencil γ is applicable to graphs α, β, then we copy the edges between source and target nodes of γ onto the sheet with pictures of α and β. In this way we obtain the joined graph of α and β.

Note that there may not be any new edge defined by a stencil γ to be drawn between any node of α and any node of β. Intuitively this means that the connection which existed between the mother nodes of α and β is now broken. A special case of this is when γ is the empty graph λ.

In this section defining recurrence systems on graphs we shall make use of this same joining mechanism, except that there joining takes place not between pairs of daughter graphs, rather between graphs generated by the previous steps of computation.

The formal definitions are as follows:

Given a concrete e-stencil γ over Σ,Δ,Δ', we define

(i) $\gamma_S = g(\mathrm{mer}(V^{\gamma}_{\Sigma_t},\gamma))$, the source part of γ;

(ii) $\gamma_T = g(\mathrm{mer}(V^{\gamma}_{\Sigma_s},\gamma))$, the target part of γ.

We say that a concrete e-stencil γ over Σ,Δ,Δ' is applicable to an ordered pair of concrete e-graphs α,β over Σ,Δ,Δ' if

(i) $V^{\alpha}_{\Sigma} \cap V^{\beta}_{\Sigma} = \phi$, i.e. the only common node of α and β is ∞;

(ii) $\gamma_S \leq \alpha$ and $\gamma_T \leq \beta$.

Note that (ii) implies $V^{\gamma} \subseteq V^{\alpha} \cup V^{\beta}$, so informally (i) and (ii) mean that α and β are disjoint and $\alpha(\beta)$ covers the source (target) part of γ including the hands. Note also that there might be two types of hands in γ_s and γ_t the "original" hands of γ plus "new" hands which were created by "breaking" of γ into γ_S and γ_T.

Let γ be a concrete e-stencil applicable to a pair of concrete e-graphs α,β. By "joining α,β by γ" we mean that we add to $\alpha \cup \beta$ the edges of $g(\gamma)-(\gamma_S \cup \gamma_T)$. Formally the joining of the ordered pair α,β of concrete e-graphs according to the concrete e-stencil γ is denoted by $\alpha \xrightarrow{\gamma} \beta$, and is defined as the concrete e-graph $(V^{\alpha} \cup V^{\beta},(E^{\alpha}-E^{\gamma_S}) \cup (E^{\beta}-E^{\gamma_T}) \cup E^{\gamma})$. Note that $\xrightarrow{\gamma}$ is an operation not a relation.

Given a set of concrete e-stencils Q, γ in Q is said to be Q-maximal for an ordered pair of concrete e-graphs α,β if γ is applicable to α,β and there is no δ in Q such that δ is applicable to α,β and $\gamma_S \cup \gamma_T < \delta_S \cup \delta_T$.

3c. Definition of graph expressions

For formal definition of both graph production systems and graph recurrence systems we need the notion of "graph expression".

Let Ω be a finite set of subsets of $[\Sigma,\Delta]_*$ and Π be a finite set of subsets of $[\Sigma_{st},\Delta]_*$. A graph expression over Σ,Δ is an abstract e-graph over Ω,Π,Δ. A graph repression A denotes a set of abstract e-graphs over Σ,Δ, written as D(A), and defined as follows.

Let α be a representant of A. A concrete graph η is a representant of an element of D(A) iff

(i) For every $n \in V^{\alpha}_{\Omega}$, i.e. for every non-environmental node of α, there exists a concrete e-graph β_n as follows.

(i1) If $n \in V^{\alpha}_{X}$ for $X \in \Omega$, then $\beta_n \in [X]$, i.e. β_n is a representant of an abstract e-graph in X.

(i2) $V_{\Sigma}^{\beta_m} \cap V_{\Sigma}^{\beta_n} = \phi$ for all m,n in V^{α}_{Ω}, $m \neq n$.

(i3) $V^{\eta} = \bigcup_{n \in V^{\alpha}_{\Omega}} V^{\beta_n}$.

(i4) $\langle V_{\Sigma}^{\beta_n} \rangle_{\eta} = \langle V_{\Sigma}^{\beta_n} \rangle_{\beta_n}$.

(ii) For every inside edge of α, say (m,n) in E_Y, $Y \in \Pi$ there exists a representant $\gamma_{m,n}$ of an abstract e-stencil from Y as follows:

(ii1) Let $Q_{m,n} = \{\gamma : [\gamma] \in Y\}$, i.e. $Q_{m,n}$ is the set of all the representants of all the abstract stencils in Y. Then $\gamma_{m,n}$ must be $Q_{m,n}$-maximal for β_m, β_n.

(ii2) Let $\delta_{m,n} = \beta_{m,n} = \beta_m \xrightarrow{\gamma_{m,n}} \beta_n$. Then $\langle V_{\Sigma}^{\delta_{m,n}} \rangle_{\eta} = \langle V_{\Sigma}^{\delta_{m,n}} \rangle_{\delta_{m,n}}$.

(iii) For all $n \in V^{\alpha}_{\Omega}$, $b \in \Delta$ and $x \in V_{\Sigma}^{\beta_n}$,

(iii1) $(\infty,x) \in E^{\eta}_{b}$ iff $(\infty,n) \in E^{\alpha}_{b}$ and $(\infty,x) \in E_{b}^{\beta_n}$.

(iii2) $(x,\infty) \in E^{\eta}_{b}$ iff $(n,\infty) \in E^{\alpha}_{b}$ and $(\infty,x) \in E_{b}^{\beta_n}$.

Note that $D(A) = \phi$ if for some edge (m,n) of α there are no β_m, β_n and $\gamma_{m,n}$ as above such that $\gamma_{m,n}$ is applicable to the pair β_m, β_n.

3d. Definition of graph production systems

A propagating graph OL-system (PGOL-system) is an ordered quintuple $G = (\Sigma,\Delta,P,C,S)$ 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 $\Delta \times [\Sigma_{st},\Delta]_+$ of connection rules;

S in $[\Sigma,\Delta]_+$ is the axiom (initial graph).

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

For U,V in $[\Sigma,\Delta]_+$ we write $U \underset{G}{\Rightarrow} V$ if there exists a graph expression W such that

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

(ii) $V \in D(W)$.

We want to stress that different productions may be used for different occurrences of the same node label in U.

Note that we add the "empty" stencil λ to every set of stencils to assure "completeness" with respect to connection rules. In this way, for every U there exists a V so that $U \underset{G}{\Rightarrow} V$. If no connection rule is applicable to a pair of right sides of productions, then the "default" stencil λ is always applicable, meaning that no new edges are created.

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

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

A PGOL-system (Σ,Δ,P,C,S) is called <u>deterministic</u> if for every U which can be generated from the axiom, i.e. $S \underset{G}{\Rightarrow}^* U$, there is exactly one V such that $U \underset{G}{\Rightarrow} V$.

The following lemma shows that by examining P, C and S we can decide whether a given PGOL-system (Σ,Δ,P,C,S) is deterministic.

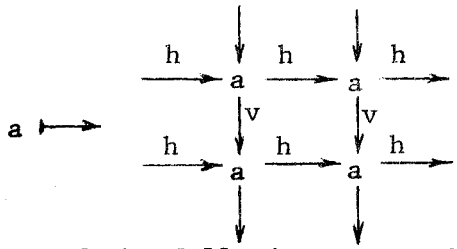
<u>Lemma 1</u>. A PGOL-system (Σ,Δ,P,C,S) is deterministic iff the following holds:

(1) For each a in Σ there is exactly one A so that $a \longmapsto A$ is in P.

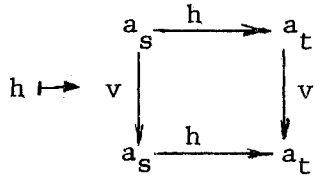
(2) Let an edge labeled by h occur in S, P or C, with its source and target labeled a_1 and a_2, respectively. Let $a_i \longmapsto A_i$ be in P and $A_i = [\alpha_i]$ for i = 1,2, and let $Q = \{\gamma : h \longmapsto [\gamma] \in C\}$. Then there must be at most one γ in Q such that γ is Q-maximal for α_1, α_2.

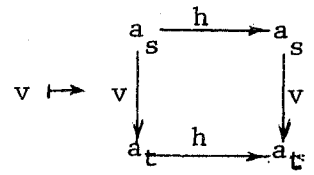
<u>Proof</u> For any U, V such that $S \underset{G}{\Rightarrow}^* U \underset{G}{\Rightarrow} V$ we have by condition (1) the unique graph expression W such that $V \in D(W)$. From condition (2) and because of the "default" stencil λ in definition of relation $\underset{G}{\Rightarrow}$ it follows that D(W) is a singleton set.

<u>Example 1</u>. In all diagrams the nodes of stencils will be labeled by a_s or a_t rather than (a,s) or (a,t) for $a \in \Delta$. Let $G = \langle\{a\},\{h,v\},P,C,S\rangle$ be a PGOL-system, where set P consists of the single production:

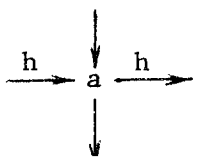


C is the set of the following connection rules:

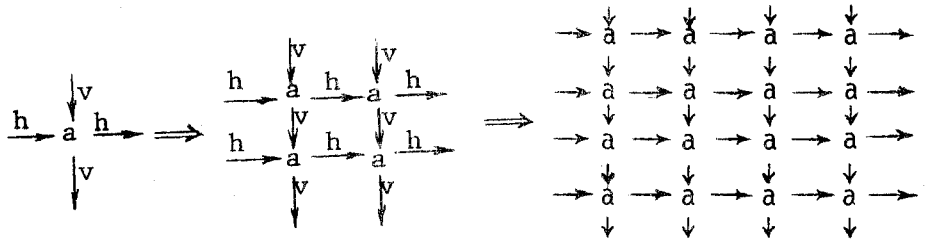




and S is the graph:



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 [13]).

4. Results on PGOL-systems

4a. Subgraph problem and reduced normal form

We want to show that for every GPOL system there exists an equivalent system without "useless" productions and connections rules. To do this we first prove a theorem which is important in itself, namely, solvability of the so called (full) subgraph problem.

Theorem 1 Given a PGOL-system G and an abstract graph A, it is recursively decidable whether A is a (full) subgraph of L(G).

Note. Theorem 1 can be extended (without changing the proof) to table PGOL-systems mentioned in section 4b.

Proof Let $G = (\Sigma,\Delta,P,C,S)$. We construct an unlabeled (concrete) directed graph $\alpha = (V,E)$ where nodes V are all the abstract graphs over Σ,Δ with no more nodes than A and for X,Y in V (X,Y) is in E iff there exists Z so that $X \underset{G}{\Rightarrow} Z$ and Y is a (full) subgraph of Z. Let $V_S = \{X \in V : X \text{ is a (full) subgraph of } S\}$. Since G is a propagating system, if $X \underset{G}{\Rightarrow} Z$ and Y is a (full) subgraph of Z, then there must exist X' and Z' so that $X' \underset{G}{\Rightarrow} Z'$, X' has no more nodes than Y and Y is a (full) subgraph of Z'. Therefore, clearly, A is a (full) subgraph of some element of L(G) iff there exists a path in α from a node in V_S to the node A. Since α is finite this is, of course, decidable. This completes the proof. □

Given a PGOL-system $G = (\Sigma,\Delta,P,C,S)$ a symbol in $\Sigma \cup \Delta$ is said to be useless if it does not occur in any element of L(G). Similarly, a production in P or connection rule in C is said to be useless if it cannot be "used" in any derivation of G, i.e. more formally:

(1) the production $a \mapsto A$ is useless if label a is useless.

(2) the connection rule $b \mapsto B$ is useless if there is no edge in an element of L(G) labeled by b and pointing from a node labeled a_1 to a node labeled a_2 so that: $a_i \mapsto [\alpha_i]$ is in P for i = 1,2, and β is Q-maximal for α_1, α_2 where β is a representant of B and $Q = \{\gamma : b \mapsto [\gamma] \in C\}$.

A PGOL-system G is called reduced if it has no useless symbols, productions or connection rules.

Theorem 2 For every PGOL-system G there effectively exists an equivalent reduced PGOL-system G'.

Proof By Theorem 1 we can determine which symbols in $\Sigma \cup \Delta$ will ever occur in L(G), so we can omit useless symbols and useless productions.

Also by Theorem 1 it is decidable which subgraphs of the form $a_1 \xrightarrow{b} a_2$ occur in L(G). For every a_1, a_2 and b we can determine which connection rules can be used. So we can determine which connection rules can be ever used and omit all the remaining. □

4b. Extension of some results on string production systems to graph production systems

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) [2].

Various special types or modifications of string OL-systems have been extensively studied, see e.g. [14] or [15] to which we refer the reader for formal definitions of the following "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, and the meanings of F and T are obvious. For C we will consider only "codings" of node labels and not edge labels, i.e. formally:

Let Σ, Σ' be two alphabets. A coding f is a function from $\Sigma \cup \{e\}$ to $\Sigma' \cup \{e\}$ such that $f^{-1}(e) = \{e\}$, extended to concrete e-graphs as follows.

For α in $(\Sigma,\Delta)_*$, $\alpha = (\{V_a\}_{a\in\Sigma}, \{E_b\}_{b\in\Delta})$, let $f(\alpha) = (\{V'_c\}_{c\in\Sigma'}, \{E_b\}_{b\in\Delta})$, where $V'_c = \bigcup_{a\in f^{-1}(c)} V_a$ for all $c \in \Sigma'$. For abstract e-graphs $f([\alpha]) = [f(\alpha)]$.

We shall use the same notation for the families of graph languages generated by various types of systems as it is common for the corresponding families of string languages (see e.g. [14]) except that the names will end in GOL rather than in OL. For example, the family of G-languages generated by propagating deterministic table GOL systems with a finite number of axioms will be denoted by DTFPGOL. Also for codings the notation is similar as in [15], namely: Let X be a combination of "operators" F, D or T. Then CXPGOL = $\{f(L): L \in$ XPGOL and f is a coding$\}$.

To conform to our formalism, in cases where we consider graphs without edge labels, we choose $\Delta = \{\#\}$. However, we will omit label # everywhere in the diagrams.

We introduce a transformation $\underline{\Phi}$ which will allow us to consider L-systems on strings as a special case of GOL systems.

$\underline{\Phi} : \Sigma^* \to (\Sigma,\{\#\})_*$, defined as $\underline{\Phi}(a_1a_2\ldots a_n) =$

$= \to a_1 \to a_2 \to \ldots \to a_n \to$, where a_i is in Σ for $i = 0,\ldots,n$.

For $L \subseteq \Sigma^*$, let $\underline{\Phi}(L) = \{\underline{\Phi}(w): w \in L\}$.

For an OL-system $G = (\Sigma,P,\sigma)$, let $\underline{\Phi}(G)$ be a PGOL-system $(\Sigma,\{\#\},P',C,\sigma')$, such that

(i) $\sigma' = \underline{\Phi}(\sigma)$,

(ii) $P' = \{a \mapsto \underline{\Phi}(w): a \to w \in P\}$

(iii) $C = \{\# \mapsto a_s \to b_t: a,b \in \Sigma\}$

Similarly let $\underline{\Phi}$ be extended to XPOL systems where X is any combination of "operators" D, F, T or C. By an CXPOL system we mean an XPOL system plus a coding.

Lemma 2. Let X be any combination of "operators" D, F, T or C. For every XPGOL system G, $\underline{\Phi}(L(G)) = L(\underline{\Phi}(G))$.

Proof. Obvious.

Lemma 3. Let X be any combination of "operators" D, F, T and C. If $L \in$ XPGOL $\cap\ \underline{\Phi}(\Sigma^+)$, then there exists $L_1 \in$ XPOL such that $\underline{\Phi}(L_1) = L$.

Proof. We will do the proof only for PGOL-systems, but it can,clearly, be modified to any XPGOL-systems. Let $G = (\Sigma',\Delta,P,C,S)$ be a reduced PGOL-system, $L(G) = L$.

Since all intermediate steps of every derivation of G are in $\underline{\Phi}(\Sigma^+)$ and G is reduced we have:

(i) $\Sigma' \subseteq \Sigma$,

(ii) $\Delta = \{\#\}$,

(iii) $\sigma' \in \underline{\Phi}(\Sigma'^+)$,

and (iv) $P \subseteq \Sigma' \times \underline{\Phi}(\Sigma'^+)$.

Now, let $C' = \{\# \mapsto x \to y : x,y \in \Sigma'\}$ and let $G' = (\Sigma',\Delta,P,C',S)$. We claim that $L(G) = L(G')$. This is so because the connections with environment are not allowed to change direction and therefore also connections of "sisters" must preserve the direction of the connections of their mothers. Otherwise, we would get a graph not in $\underline{\Phi}(\Sigma^+)$.

G' is in $\underline{\Phi}$(POL)so there is a POL-system $\bar{G}$ so that $\underline{\Phi}(\bar{G}) = G'$. By Lemma 2 $\underline{\Phi}(L(\bar{G})) = L(\underline{\Phi}(G'))$. Since $L(\underline{\Phi}(G')) = L$ the proof is completed. □

Lemma 4. Let X and Y be any combinations of "operators" D, F, T and C. If $L \in$ XPOL-YPOL, then $\underline{\Phi}(L) \in$ XPGOL-YPGOL.

Proof. Let G be an XPOL-system generating L. By Lemma 2 $\underline{\Phi}(L(G)) = L(\underline{\Phi}(G))$. Since $\underline{\Phi}(G)$ is an XPGOL-system we have $\underline{\Phi}(L) \in$ XPGOL. Let assume that $\underline{\Phi}(L)$ is also YPGOL. By Lemma 3 there is G' in YPOL such that $\underline{\Phi}(L(G')) = \underline{\Phi}(L)$. Since $\underline{\Phi}$ is one-to-one we have $L(G') = L$ and L is in YPOL which is a contradiction. □

Corollary 1. Let X and Y are as before. If XPOL and YPOL are incomparable then XPGOL and YPGOL are incomparable. If XPGOL $\subseteq$ YPGOL and XPOL $\subsetneq$ YPOL then XPGOL $\subsetneq$ YPGOL.

Proof. Immediately by Lemma 4. Note, however, that from Lemma 4 does not follow that the inclusion XPOL $\subseteq$ YPOL implies the inclusion XPGOL $\subseteq$ YPGOL.

A number of results about various families of languages generated by modified or restricted string OL-systems is summarized in the diagram of Fig.1. The meaning of the diagram is the following. If two nodes, say X and Y, are connected by an edge the node X being below the node Y, then $X \subsetneq Y$. If these two nodes are connected by a broken edge then X and Y are incomparable. The results summarized in Fig.1 are either obvious, or given in [14], [15], Corollary 2 or Lemma 7.

Lemma 5. CDPOL is incomparable to both POL and FPOL.

Proof. The language $\{a^{2^n} b\, a^{3^n} : n \geq 1\}$ is clearly in CDPOL-FPOL. The language $\{a\}^+ \{b\}^+$ is clearly in POL but by Lemma 3 from [9] it is not in CDPOL. □

Lemma 6. Let L_1 be $\{aaa\} \cup \{a^{2^n} : n \geq 2\}$. Then $L_1 \in$ CDPOL-TFPOL.

Proof. Let G be the DPOL-system $(\{a_1, a_2\}, \{a_1 \to a_1a_1, a_2 \to a_1\}, a_1a_2a_2)$ let $h(a_i) = a$ for $i = 1,2$. Obviously, $h(L(G)) = L_1$ and clearly $L_1 \notin$ TFPOL. □

Lemma 7. Let h be the homomorphism on $\{a,b\}^*$ defined by $h(a) = a$, $h(b) = \varepsilon$ (where ε is the empty string), and let $L_2 = h^{-1}(\{a^{2^n} : n \geq 1\})$.

$L_2 \in$ TPOL (TFPOL) but $L_2 \notin$ FPOL, CPOL, CFPOL.

Proof. It is shown in [19] that $L_2 \in$ TPOL - EOL and all three families FPOL, CPOL and CFPOL are included in EOL. □

Corollary 2. Any of the families CDPOL, CFDPOL, CPOL and CFPOL is incomparable to both TPOL and TFPOL.

Proof. Follows immediately from Lemmas 6 and 7 and obvious inclusions. □

Now, we will extend all the results from Fig.1 to graph-languages.

Theorem 3. The results summarized in the diagram of Fig.2 hold.

Proof. All the inclusions (but not necessary proper inclusions) from Fig.2 are obvious. Since all the corresponding inclusions in Fig.1 are proper, by Corollary 1, also all the inclusions in Fig.2 are proper. Each incomparability result of Fig.2 follows from the corresponding result of Fig.1 by Corollary 1. □

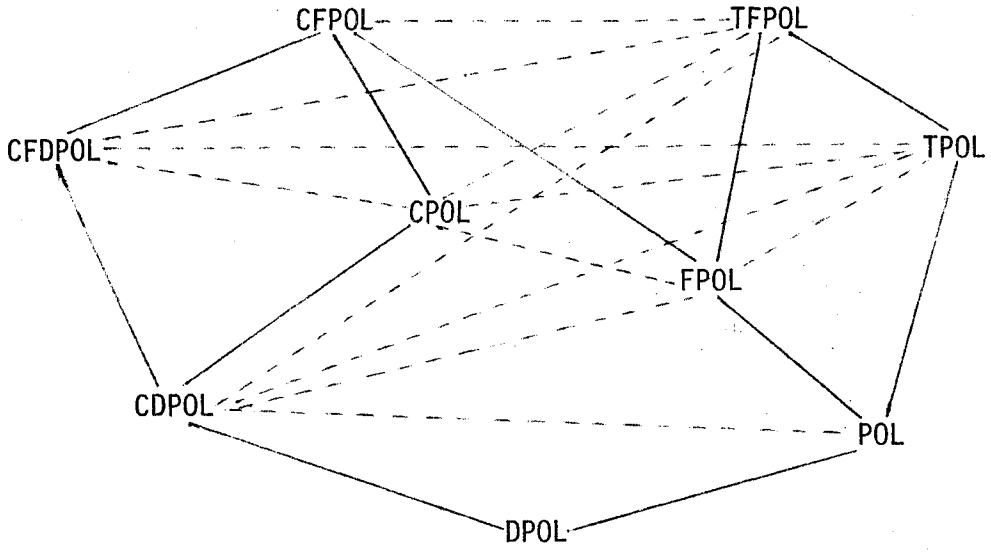


Figure 1

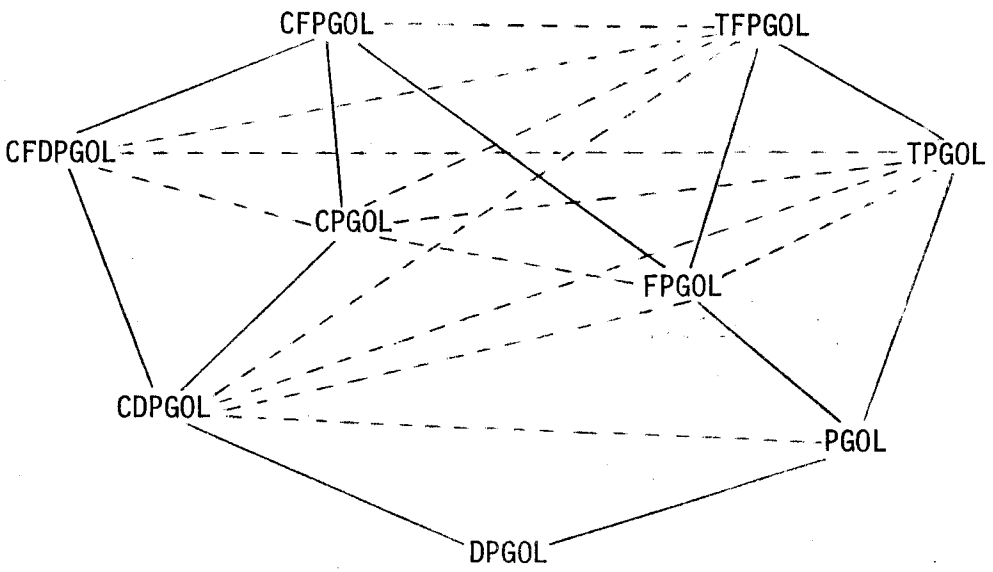


Figure 2

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

Theorem 4. 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".

Proof. It is quite easy but rather tedious to prove this result and we are leaving it to the reader. □

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.

5. Recurrence systems on graphs

5a. Definition of stencil expressions

For definition of recurrence systems we need not only graph expressions but also stencil expressions.

Stencil expressions of two kinds are used. The first kind (I) of stencil expressions are graphs over stencils, and are used to produce larger and larger stencils from previously defined stencils. In other words, a stencil expression of kind I is an abstract e-graph over Ω,Π, where both the set of node labels Ω and the set of edge labels Π are finite sets of sets of abstract e-stencils.

The second kind (II) of stencil expressions are stencils over graphs, and are used to produce stencils from larger and larger graphs. A stencil expressions of kind II is an abstract e-stencil over Ω,Π, where Ω is a finite set of sets of abstract e-graphs and Π is a finite set of sets of abstract e-stencils.

A stencil expression A denotes the set of stencils D(A) which is defined analogously as the set of graphs defined by graph expressions, with the following modifications.

The formal definitions are as follows:

Case I: Let α be a representant of A. A concrete e-stencil η is a representant of an element of D(A) where A is a stencil expression over Ω, Π iff :

(i) For every $n \in V_{\Omega}^{\alpha}$ there exists β_n as follows:

(ii) If $n \in V_{X}^{\alpha}$ for $X \in \Omega$, then $[\beta_n]$ is in X.

(i2) $V^{\beta_m}_{\Sigma_{st}} \cap V^{\beta_n}_{\Sigma_{st}} = \phi$ for all m,n in V^{α}_{Ω}, $m \neq n$.

(i3) $V^{\eta} = \bigcup_{n \in V^{\alpha}_{\Omega}} V^{\beta_n}$.

(i4) $\langle V^{\beta_n}_{\Sigma_{st}} \rangle_{\eta} = \langle V^{\beta_n}_{\Sigma_{st}} \rangle_{\beta_n}$

(ii) For every inside edge of α, say $(m,n) \in E_{\gamma}$, $\gamma \in \Pi$ there exists a representant $\gamma_{m,n}$ of an abstract e-stencil from γ as follows:

(iii1) Let $Q_{m,n} = \{\gamma : [\gamma] \in \gamma\}$. Then $\gamma_{m,n}$ is $Q_{m,n}$-maximal for β_m, β_n.

(ii2) Let $\delta_{m,n} = g(\beta_m) \xrightarrow{\gamma_{m,n}} g(\beta_n)$. Then

$$g(\langle V^{\delta_{m,n}}_{\Sigma} \rangle_{\eta}) = \langle V^{\delta_{m,n}}_{\Sigma} \rangle_{\delta_{m,n}} .$$

(iii) For all n in V^{α}_{Ω}, b in Δ and x in $V^{\beta_n}_{\Sigma}$

(iii1) $(\infty,x) \in E^{\eta}_{b}$ iff $(\infty,n) \in E^{\alpha}_{b}$ and $(\infty,x) \in E^{\beta_n}_{b}$,

(iii2) $(x,\infty) \in E^{\eta}_{b}$ iff $(n,\infty) \in E^{\alpha}_{b}$ and $(x,\infty) \in E^{\beta_n}_{b}$.

Case II: We will show only differences from Case I.

(i1) If $n \in V^{\alpha}_{(X,s)} \cup V^{\alpha}_{(X,t)}$ for $X \in \Omega$, then $[\beta_n]$ is in X.

(i2) $V^{\beta_m}_{\Sigma} \cap V^{\beta_n}_{\Sigma} = \phi$ for all $m,n \in V^{\alpha}_{\Omega_{st}}$, $m \neq n$.

(i3) $V^{\eta}_{(a,x)} = \bigcup_{n \in \Omega_x} V^{\beta_n}_{a}$, for all $a \in \Sigma$ and $x \in \{s,t\}$.

(i4) $g(\langle V^{\beta_n}_{\Sigma} \rangle_{\eta}) = \langle V^{\beta_n}_{\Sigma} \rangle_{\beta_n}$.

(ii2) Let $\delta_{m,n} = \beta_m \xrightarrow{\gamma_{m,n}} \beta_n$. Then the same holds as in Case I.

5b. Definition of recurrence systems on graphs

Analogously to string recurrence systems [16], 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.

Let $\underline{d} = \{i:1 \leq i \leq d\}$ for every positive integer d.

A graph recurrence system is an ordered 8-tuple

$S = (\Sigma,\Delta,\Gamma,\Theta,d,A,F,\omega)$ where:

Σ is an alphabet of node labels.

Δ is an alphabet of edge labels.

Ω is a finite set of graph variables.

Π is a finite set of stencil variables.

d is a positive integer, the depth of S.

A is a function assigning to each (x,n) in $(\Gamma \cup \Theta) \times \underline{d}$ a finite set $A_{x,n}$, where $A_{x,n} \subseteq [\Sigma,\Delta]_*$ for $x \in \Gamma$ and $A_{x,n} \subseteq [\Sigma_{st},\Delta]_*$ for $x \in \Theta$. A is the axiom function.

F is a function assigning to each x in $\Gamma \cup \Theta$ a finite set F_x where $F_x \subseteq [(\Gamma \times \underline{d}) \cup \Sigma, \Theta \times \underline{d}, \Delta]_*$ for each x in Γ and $F_x \subseteq [((\Gamma \times \underline{d}) \cup \Sigma)_{st}, \Theta \times \underline{d}, \Delta]_* \cup [\Theta \times \underline{d}, \Theta \times \underline{d}, \Delta]_*$ for each x in Θ. The set $\{(x,0) = q:x \in \Gamma \cup \Theta, q \in F_x\}$ is the set of recurrence formulae and we will write them in the more usual form where each (x,i) in $(\Gamma \cup \Theta) \times \underline{d}$ is replaced by x_{n-i}.

ω in Γ is the distinguished variable.

Now, we give the usual computational interpretation of recurrence systems. Let $S = (\Sigma,\Delta,\Gamma,\Theta,A,F,\omega)$ be a graph recurrence system. For x in $\Gamma \cup \Theta$ and $n \geq 0$ we define $L_{x,n}(S)$ as follows.

If $n < d$, then $L_{x,n}(S) = A_{x,n+1}$.

Let $F_{x,n}$ be the set of graph (stencil) expressions obtained from F_x by replacing every label (z,i) in $(\Gamma \cup \Theta) \times \underline{d}$ by the set of graphs (stencils) $L_{z,n-i}$ and every label from $\Gamma_2 \times \underline{d}$ by the label $(L_{z,n-i})_q$ for $q \in \{s,t\}$. If $n \geq d$ then $L_{x,n}(S) = \bigcup_{B \in F_{x,n}} D(B)$.

Finally, the graph language generated by recurrence system S is denoted by L(S) and defined as $L(S) = \bigcup_{n=1}^{\infty} L_{\omega,n}(S)$.

Example 2. Now, we give a recurrence system for the graph language generated by the PGOL-system in Example 1.

Let $S = (\{a\},\{h,v\},\{Q\},\{H,V,\bar{H},\bar{V}\},1,A,F,Q)$ be a graph recurrence system with functions A,F given in Figure 3. Each entry in the table of Figure 3 is a singleton set.

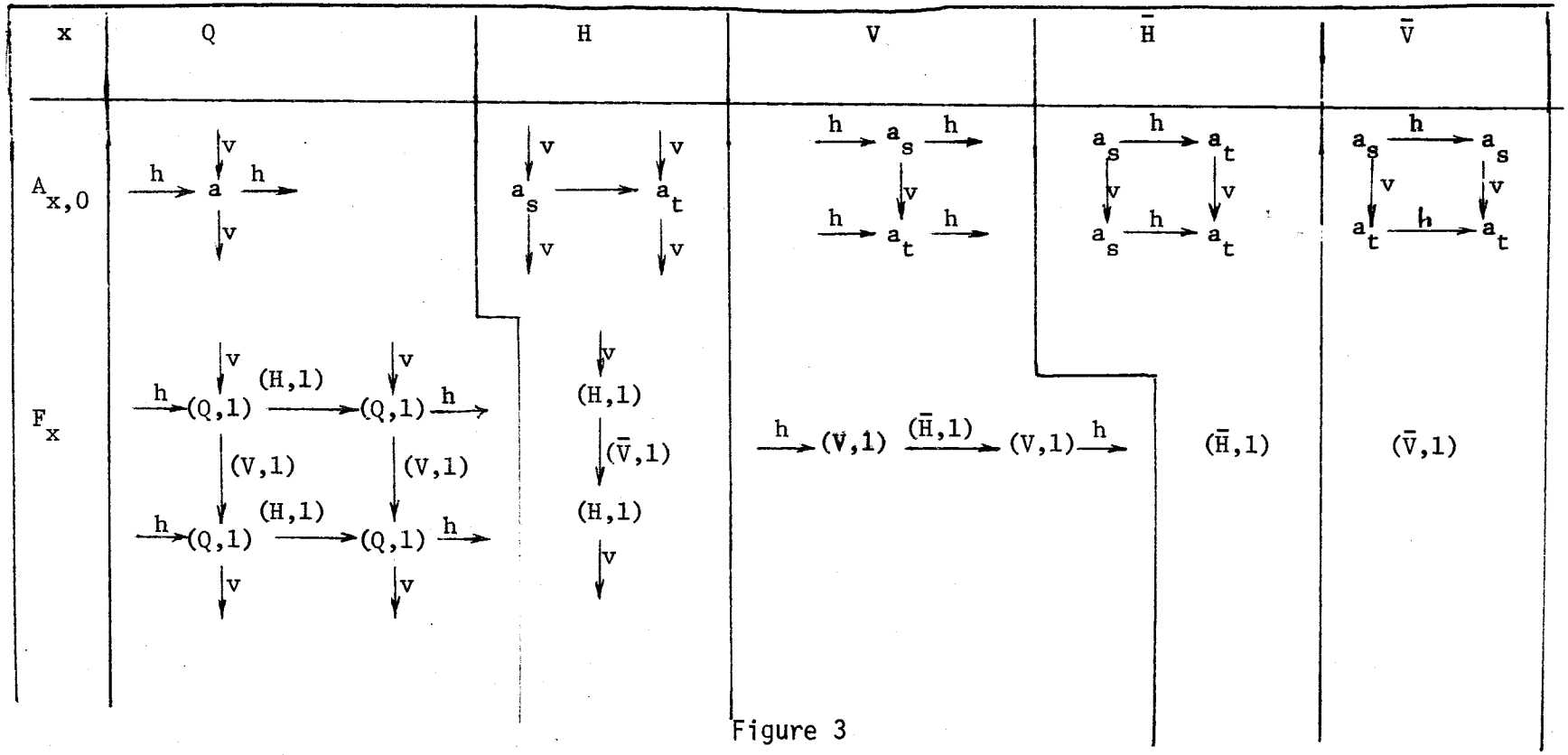
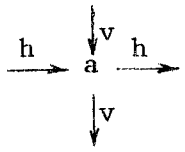
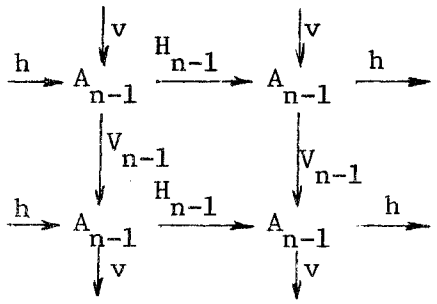
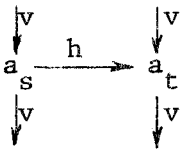


Figure 3

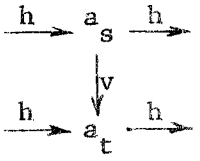
Using the more usual notation our recurrence system is as follows, where $\bar{H}$ and $\bar{V}$ are considered to be "constant" (sets of) stencils. According to the formal definition, initial values for each variable is a set, but in this case all these sets $(A_0, H_0, V_0, \bar{H}, \bar{V})$ are singletons and we write them without curly brackets.

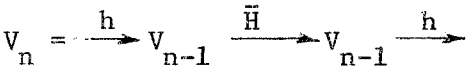
$A_0 =$ 

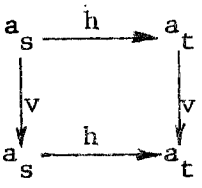
$A_n =$ 

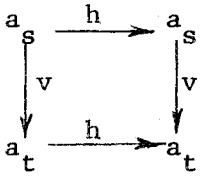
$H_0 =$ 

$H_n =$ 

$V_0 =$ 

$V_n =$ 

$\bar{H} =$ 

$\bar{V} =$ 

Let $S = (\Sigma,\Delta,\Gamma,\Theta,d,A,F,\omega)$ be a recurrence system. If $A_{x,n}$ is a singleton or empty set for every (x,n) in $\Gamma \times \underline{d}$ and there is only one recurrence formula for every x in Γ, then we call S a pseudo-deterministic system. This property does not necessary imply that $L_{x,n}(S)$ has no more than one element for every x in Γ and $n \geq 0$.

The recurrence system from Example 2 is pseudo-deterministic.

6. Relationship of graph production and recurrence systems and decidability results

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.

Since neither PGOL-systems nor λ-free recurrence systems allow erasing we clearly have the following.

Theorem 5. The membership problem is decidable for both PGOL-systems and λ-free recurrence systems on graphs.

The following theorem is a generalization of a similar result for string-systems [16].

Theorem 6. For every PGOL-system there effectively exists an equivalent λ-free recurrence system, i.e. a λ-free recurrence systems generating the same graph language.

Proof. Given a PGOL-system $G = (\Sigma,\Delta,P,C,S)$ we construct recurrence system

$$Q = (\Sigma,\Delta,\bar{\Sigma} \cup \{\omega\},\bar{\Delta},1,A,F,\omega) \text{ where } \omega \notin \Sigma \cup \Delta$$

$$\bar{\Sigma} = \{\bar{a}:a \in \Sigma\},\ \bar{\Delta} = \{\bar{b}:b \in \Delta\}$$

Let μ be a mapping on graphs or stencils which changes any label z in $\Sigma \cup \Delta \cup \Sigma_{st} \cup \Delta_{st}$ to $(\bar{z},1)$. Let

$$A_\omega = \{S\},$$

$$A_a = \{W:a \mapsto W \in P\} \quad \text{for each } a \text{ in } \Sigma,$$

$$A_b = \{W:b \mapsto W \in C\} \quad \text{for each } b \text{ in } \Delta,$$

$F_\omega = \{\mu(S)\}$

$F_{\bar{a}} = \{\mu(W): a \mapsto W \in P\}$ for each a in Σ,

$F_{\bar{b}} = \{\mu(W): b \mapsto W \in C\}$ for each b in Δ.

It can be verified by induction that $L_{a,n} = \{Y: X \in A_a, X \overset{n}{\underset{G}{\Rightarrow}} Y\}$ for all $a \in \Sigma$, and $L_{\omega,n} = \{Y: S \overset{n}{\underset{G}{\Rightarrow}} Y\}$. Thus $L(G) = L(Q)$.

Note that using the construction of the above proof we can obtain another recurrence system equivalent to the PGOL-system from Example 1. It is essentially different from the one shown in Example 2, namely, the stencils H_n and V_n are much larger (unnecessarily) in this general construction which uses the stencil expressions of type II.

In Example 2 a more natural ad hoc construction is shown using stencil expression of type I.

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 (compare also with Theorem 1) 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 7. The emptiness problem is undecidable for (λ-free, pseudo-deterministic) graph recurrence systems.

Proof. Let $A = x_1,\ldots,x_k$ and $B = y_1,\ldots,y_k$ where $x_i, y_i \in \Sigma^+$ for $1 \leq i \leq k$ be the lists in an instance of Post Correspondence Problem, see [17]. Let $x_i = x_{i1}x_{i2}\ldots x_{ip_i}$, $y_i = y_{i1}y_{i2}\ldots y_{iq_i}$ where $x_{ij}, y_{ij} \in \Sigma$ for $i = 1,\ldots,k$ and $j = 1,\ldots,p_i$ (q_i). Let $K = \{a_1,\ldots,a_k\}$ be an alphabet such that $\Sigma \cap K = \phi$.

We will construct graph recurrence system S which generates the empty graph language iff the given instance of Post Correspondence Problem has no solution.

Construct $S = (\Sigma \cup K,\{\#,v\},\Gamma,\Theta,1,A,F,Q)$ where $\Gamma = \{Q,A,B,X^1,\ldots,X^k,Y^1,\ldots,Y^k\}$, $\Theta = \{W,U,V,L,R,M,H\}$ and A, F are defined as follows, using the usual notation rather than strictly our formalism. Note that X^i, Y^i for $i = 1,\ldots,k$ and W,X,Y,L,R,M are "constant" graphs or stencils. Edge label # is omitted in all diagrams as usually.

$$X^i = \{ \rightarrow x_{i1} \rightarrow x_{i2} \rightarrow \ldots \rightarrow x_{ip_i} \rightarrow \} \quad \text{for } i = 1,\ldots,k$$

$$Y^i = \{ \rightarrow y_{i1} \rightarrow y_{i2} \rightarrow \ldots \rightarrow y_{iq_i} \rightarrow \} \quad \text{for } i = 1,\ldots,k$$

$$A_0 = \left\{ \begin{array}{l} \rightarrow a_1 \rightarrow x_{11} \rightarrow \ldots \rightarrow x_{1p_1} \rightarrow \\ \rightarrow a_2 \rightarrow x_{21} \rightarrow \ldots \rightarrow x_{2p_2} \rightarrow \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \\ \rightarrow a_k \rightarrow x_{k1} \rightarrow \ldots \rightarrow x_{kp_k} \rightarrow \end{array} \right\}$$

(A_0 is a singleton set containing one graph which is not connected.)

$$A_n = \begin{array}{l} \rightarrow a_1 \xrightarrow{W} A_{n-1} \xrightarrow{U} X^1 \rightarrow \\ \rightarrow a_2 \xrightarrow{W} A_{n-1} \xrightarrow{U} X^2 \rightarrow \\ \quad\vdots \quad\quad\quad\quad\quad \vdots \\ \rightarrow a_k \xrightarrow{W} A_{n-1} \xrightarrow{U} X^k \rightarrow \end{array}$$

(On the right side of this recurrence formula there is again one disconnected graph.)

$$B_0 = \left\{ \begin{array}{l} \rightarrow a_1 \rightarrow y_{11} \rightarrow \ldots \rightarrow y_{1q_1} \rightarrow \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \\ \rightarrow a_k \rightarrow y_{k1} \rightarrow \ldots \rightarrow y_{kq_k} \rightarrow \end{array} \right\}$$

$$B_n = \begin{matrix} \rightarrow a_1 \xrightarrow{W} B_{n-1} \xrightarrow{V} Y^1 \rightarrow \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ \rightarrow a_k \xrightarrow{W} B_{n-1} \xrightarrow{V} Y^k \rightarrow \end{matrix}$$

$$Q_0 = \phi, \; Q_n = \begin{matrix} A_{n-1} \\ \downarrow H_{n-1} \\ B_{n-1} \end{matrix}$$

$$W = \left\{ (a_i)_s \rightarrow (a_1)_t, (a_2)_t, \ldots, (a_k)_t \; : \; 1 \le i \le k \right\}$$

$$U = \left\{ (x_{1p_1})_s, (x_{2p_2})_s, \ldots, (x_{kp_k})_s \rightarrow (x_{i1})_t \; : \; 1 \le i \le k \right\}$$

$$V = \left\{ (y_{1q_1})_s, (y_{2q_2})_s, \ldots, (y_{kq_k})_s \rightarrow (y_{i1})_t \; : \; 1 \le i \le k \right\}$$

$$M = \left\{ \begin{matrix} \rightarrow c_s \rightarrow d_t \rightarrow \\ \quad v\downarrow \quad\quad v\downarrow \\ \rightarrow c_s \rightarrow d_t \rightarrow \end{matrix} \; : \; c,d \in \Sigma \cup K \right\}$$

$$L = \left\{ \begin{matrix} \rightarrow (a_i)_s \rightarrow \\ \quad v\downarrow \\ \rightarrow (a_i)_t \rightarrow \end{matrix} \; : \; 1 \le i \le k \right\}$$

$$R = \left\{ \begin{array}{ccccc} \rightarrow (x_{i1})_s \rightarrow & (x_{i2})_s \rightarrow \ldots \rightarrow & (x_{ip_i})_s \rightarrow \\ \downarrow v & \downarrow v & \downarrow v \\ \rightarrow (x_{i1})_t \rightarrow & (x_{i2})_t \rightarrow \ldots \rightarrow & (x_{ip_i})_t \rightarrow \end{array} : 1 \leq i \leq k \right\}$$

$$H_0 = \rightarrow L \xrightarrow{M} R \rightarrow$$

$$H_n = \rightarrow L \xrightarrow{M} H_{n-1} \xrightarrow{M} R \rightarrow$$

Clearly, $Q_n = \phi$ for $n > 0$ iff no stencil from H_{n-1} is applicable to a pair of graphs from A_{n-1} and B_{n-1} respectively. It is easy to verify that this happens iff there is no sequence of integers $i_1,\ldots,i_{n-1}$ such that $x_{i_1}\ldots x_{i_{n-1}} = y_{i_1}\ldots y_{i_{n-1}}$. Thus, $L(S) = \phi$ iff the instance of Post Correspondence Problem with lists A, B has no solution. □

Lemma 8. Let F be any family of graph generating system with decidable membership problem, e.g. PGOL-systems or λ-free graph recurrence systems. If subgraph problem is recursively decidable for family F then emptiness problem is also decidable for F.

Proof. Assume that subgraph problem is decidable for F. Given a system S in F over node label alphabet Σ we can check for every a in Σ whether the graph with single node labeled by a is a subgraph of some element of L(S). Clearly $L(S) = \phi$ iff the answer is "no" for every a in Σ and λ is not in L(S) which is also decidable. □

Theorem 8. The emptiness problem is recursively decidable for PGOL-systems.

Proof. By Theorem 7 and Lemma 8. □

Theorem 9. The subgraph problem is recursively undecidable for (λ-free) graph recurrence systems.

Proof. By Theorem 7 and Lemma 8. □

From the undecidability of the equivalence problem for (string) POL-systems [18] and Theorem 6 it follows immediately

Theorem 10. Given two PGOL-systems G_1 and G_2 (or two graph recurrence systems S_1 and S_2) it is recursively undecidable whether $L(G_1) = L(G_2)$ ($L(S_1) = L(S_2)$).

7. Prospects for further research

The definition of graph generating systems without interactions (PGOL-systems) can be readily extended to system with interactions in the following way.

A propagating graph system with interactions (PGIL-system) is a quintuple $G = (\Sigma,\Delta,P,C,S)$ where Σ,Δ,C,S are as in a PGOL-system and P is a finite subset of $[\Sigma,\Delta]^I_+ \times [\Sigma,\Delta]_+$, where $[\Sigma,\Delta]^I$ is the set of abstract graphs over Σ,Δ with exactly one occurrence of a label distinguished (changed) by underlining it.

The definition of relation $\underset{G}{\Rightarrow}$ is very similar as for PGOL, only when replacing nodes by subgraphs we consider their "context". For U,V in $[\Sigma,\Delta]_+$, we write $U \underset{G}{\Rightarrow} V$ if there exists a graph expression W such that

(i) W is obtained by relabeling of U so that:

(i1) Each occurrence of every node label, say a, is replaced by {A} for some $Q \mapsto A$ in P such that Q is a subgraph of U', U' being obtained from U by underlining the currently considered occurrence of a.

(i2) Each occurrence of an edge label, say b, at an inside edge is replaced by the set of abstract stencils $\{B;b \mapsto B \in C\} \cup \{\lambda\}$.

(ii) $V \in D(W)$.

Note that (i2) and (ii) are exactly as in the case of PGOL systems.

Because these systems are nonerasing, it is clear that the membership problem for them is decidable (analogous result to Theorem 5). On the contrary, both emptiness and subgraph problems are undecidable for PGIL-system. The undecidability of their emptiness problem could be

transferred to the known result of undecidability of emptiness problem for context-sensitive languages [17]. The undecidability of subgraph problem then follows by Lemma 8.

A further natural extension of graph L-systems is to omit the restriction to propagating systems, i.e. to allow the erasure of nodes. Concomitantly the graph recurrence systems may be extended by omitting the λ-free restriction. These extensions would be easily implemented by defining the effect of erasing of a node as breaking of connections. However, using this definition, string systems with erasing, as considered in the literature [2], would not be special cases of graph systems with erasing. A definition which fulfills this requirement can be given, but it would be outside the scope of the present paper.

REFERENCES

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

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

3. Mayoh, B.H.: Mathematical Models for Cellular Organisms. Dept. of Comp. Sci., Aarhus Univ., Denmark, Rep. No. DAIMI PB-12, 38 pp., 1973.

4. Mayoh, B.H.: Multidimensional Lindenmayer Organisms. In: L Systems (edited by G. Rozenberg and A. Salomaa), Lect. Notes in Comp. Sci., Vol.15, pp.302-326, Heidelberg: Springer-Verlag, 1974.

5. von Neumann, J. (completed by A.W. Burks): Theory of Self-Reproducing Automata. Urbana: Univ. of Illinois Press, 1966.

6. Codd, E.F.: Cellular Automata. New York: Academic Press, 1968.

7. Pfaltz, J.L., Rosenfeld, A.: Web Grammars. Proc. Int. Joint Conf. on Artificial Intelligence, Washington, pp.609-619, 1969.

8. Yamada, H., Amoroso, S.: Tessellation Automata. Information and Control 14, 299-317 (1969).

9. Milgram, D., Rosenfeld, A.: Array Automata and Array Grammars. Proc. IFIP Congress, TA-2, pp.166-173, 1971.

10. Rosenfeld, A., Milgram, D.: Web Automata and Web Grammars. Machine Intelligence 7, pp.307-324 (edited by B. Meltzer and D. Mitchie) Edinburgh: Univ. Press, 1972.

11. Ehrig, H., Pfender, M., Schneider, H.J.: Graph-grammars: An Algebraic Approach. 14th Annual Symp. on Switching and Automata Theory, pp.167-180, 1973.

12. Carlyle, J.W., Greibach, S.A., Paz, A.: A two-dimensional Generating System Modeling Growth by Binary Cell Division. 15th Annual Symp. on Switching and Automata Theory, pp.1-12, 1974.

13. Lindenmayer, A., Culik, K. II: Graph Systems and Languages for Multi-dimensional Cellular Development. Paper in preparation.

14. Nielsen, M., Rozenberg, G., Salomaa, A., Skyum, S.: 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, 50 pp., 1974.

15. Culik, K. II, Opatrny, J.: Literal Homomorphisms of OL-languages. Int. J. Comp. Math. (in press); extended abstract in Proc. 1974 Conf. on Biologically Motivated Automata Theory, pp.50-53, Long Beach (Calif.): IEEE Comp. Soc., 1974.

16. Herman, G.T., Lindenmayer, A., Rozenberg, G.: Descriptions of Developmental Languages Using Recurrence Systems. Math. Systems Theory (in press).

17. Hopcroft, J.E., Ullman, J.D.: Formal Languages and Their Relation to Automata. Reading (Mass): Addison-Wesley, 1969.

18. Blattner, M.: The Unsolvability of the Equality Problem for Sentential Forms of Context-free Grammars. J. Computer and System Sci. 7, 463-486, 1973.

19. Herman, G.T.: Closure Properties of Some Families of Languages Associated with Biological Systems. Information and Control 24, 101-121, 1974.