

AD-A060 380

MARYLAND UNIV COLLEGE PARK COMPUTER SCIENCE CENTER

F/G 9/4

CELLULAR GRAPH ACCEPTORS, 5: CLOSURE PROPERTIES OF CELLULAR D-G--ETC(U)

JUL 78 A WU

AFOSR-77-3271

UNCLASSIFIED

TR-681

AFOSR-TR-78-1372

NL

1 of 1

AD
A060 380



END

DATE

FILMED

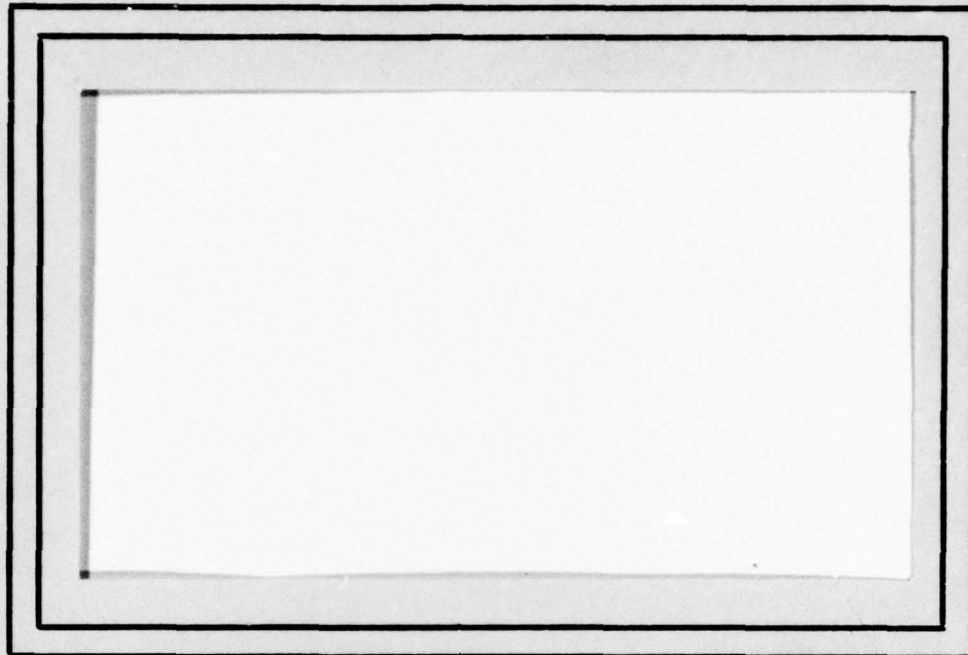
1-79

DDC

LEVEL II

2
B.S.

ADA060380



DDC FILE COPY

ADA 2 A050 360

COMPUTER SCIENCE
TECHNICAL REPORT SERIES



UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND

20742

78 10 16 135

DDC
RECEIVED
OCT 26 1978
D



AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTES OF ADDENDUM TO DDC
This technical report has been reviewed and is
approved for public release IAW AFR 190-12 (7b).
Distribution is unlimited.
A. D. BLOSE
Technical Information Officer

ADDITIONAL INFO	
DTIC	DTIC Section ☒
DDC	DDC Section ☐
UNCLASSIFIED	☐
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	AVAIL. and/or SPECIAL
A	

LEVEL II

2

TR- 681
AFOSR-77-3271

July 1978

44-4058792

CELLULAR GRAPH ACCEPTORS, 5: CLOSURE PROPERTIES OF CELLULAR d-GRAPH LANGUAGES

Angela Wu
Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

Cellular d-graph languages are shown to be closed under set theoretic operations, including finite union and intersection; and under "geometric" operations, including permutation of arc end numbering, concatenation, closure, and formation of line graphs. Determinism is preserved under the set-theoretic operations; but under the geometric operations, determinism is known to be preserved only when the languages are also predicates.

The support of the U.S. Air Force Office of Scientific Research under Grant AFOSR-77-3271 is gratefully acknowledged, as is the help of Kathryn Riley in preparing this paper.

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

DDC
RECEIVED
OCT 26 1978
D

AD A060380

DDC FILE COPY

78 10 16 135

1. Introduction

Cellular d-graph acceptors were defined and studied in [1-4]. Given a cellular d-graph acceptor $M = (\Gamma, M, H)$ with a distinguished node having a special mark, we say that M accepts Γ if the automaton at the distinguished node enters a final state after finitely many steps. For a given finite state automaton $M = (Q_I, Q, \delta, F)$, where δ is a function from $Q \times D^d \times Q^d$ into Q , $D = \{1, 2, \dots, d\}$, let $C(M) = \{M \mid M = (\Gamma, M, H) \text{ is a cellular d-graph acceptor with a distinguished node}\}$ be the class of cellular d-graph acceptors determined by M. Whenever no confusion can arise, we shall refer to $C(M)$ as the cellular d-graph acceptor of M. The language of d-graphs accepted by $C(M)$ is the set $L(M) = \{\Gamma \mid M = (\Gamma, M, H) \in C(M) \text{ accepts } \Gamma\}$. A d-graph Γ is accepted by $C(M)$ iff. $\Gamma \in L(M)$. The class of languages accepted by (deterministic) cellular d-graph acceptors is called the class of (deterministic) cellular d-graph languages, $(D)CdL$.

If we call the final states F of M accepting states and in addition M has another special subset R of states such that $R \subseteq Q$ and $R \cap F = \emptyset$ called the rejecting states, M in this case is denoted by a 5-tuple (Q_I, Q, δ, F, R) . We say that $M = (\Gamma, M, H)$ rejects Γ if the automaton at the distinguished node enters a rejecting state after finitely many steps. Let $L'(M) = \{\Gamma \mid M = (\Gamma, M, H) \in C(M) \text{ rejects } \Gamma\}$. A d-graph Γ is rejected by $C(M)$ iff. $\Gamma \in L'(M)$. $C(M)$ recognizes a d-graph

language L iff. $C(M)$ accepts every d-graph $\Gamma \in L$ and $C(M)$ rejects every d-graph $\Gamma \notin L$. L is sometimes called a cellular d-graph predicate, and $C(M)$ is called a cellular d-graph recognizer. Every cellular d-graph predicate is a cellular d-graph language.

For any d-graph Γ , $U(\Gamma)$ denotes the underlying graph of Γ , and for any $\gamma \in G_L^d = \{\gamma | \gamma = U(\Gamma) \text{ for some d-graph } \Gamma \text{ over } L\}$, $U^{-1}(\gamma) = \{\Gamma | \gamma = U(\Gamma)\}$. $C(M)$ accepts a node-labelled graph γ if $C(M)$ accepts some d-graph in $U^{-1}(\gamma)$. The acceptance is strong if $C(M)$ accepts every d-graph in $U^{-1}(\gamma)$. $C(M)$ accepts (strongly) a language $L \subseteq G_L^d$ when $\gamma \in L$ iff. $C(M)$ accepts γ (strongly). $C(M)$ rejects a node-labelled graph γ if $C(M)$ rejects every d-graph in $U^{-1}(\gamma)$. $C(M)$ recognizes a predicate $L \subseteq G_L^d$ iff. $C(M)$ accepts every $\gamma \in L$ and $C(M)$ rejects every $\gamma \notin L$. The recognition is strong if the acceptance is strong.

It should be pointed out here that " $C(M)$ accepts γ strongly" means that acceptance by $C(M)$ does not depend on how the arc ends of γ are labelled. However, each transition of M may depend on the neighbor vector, so that each $M = (\Gamma, M, H)$ for $\Gamma \in U^{-1}(\gamma)$ need not be a weak cellular d-graph automaton as defined in [1]. Conversely, a weak cellular d-graph automaton M has a transition function that is independent of the neighbor vector H , but acceptance of a graph by M may depend on the numbering of the arc ends; thus the acceptance of the underlying graph is not necessarily strong.

In the definition of a cellular d-graph acceptor with a distinguished node D , D may be any non- $\#$ node of the d-graph. In fact, in all the cellular d-graph automata defined in [1-4], the position of D does not affect whether the d-graph is accepted or not. Of course the speed of acceptance can depend on where D is placed, for example, when D is at a central point of the d-graph [4]. In this paper, we are not concerned with speed. We assume that all the cellular d-graph automata in this paper have a distinguished node.

We presented some deterministic cellular d-graph languages and predicates in [2-4]. In this paper we will investigate closure properties of the cellular d-graph languages under various set-theoretic and geometric operations.

2. Set-theoretic operations

Proposition 1: A finite intersection of (deterministic) cellular d-graph languages (predicates) is a (deterministic) cellular d-graph language (predicate).

Proof: Let $M_1, \dots, M_n$ be finite state acceptors such that the cellular d-graph acceptors $C(M_1), \dots, C(M_n)$ accept the Cdl's $L_1, \dots, L_n$. Define M as follows: the states of M are n -tuples in $Q_1 \times \dots \times Q_n$ and the accepting states of M are $F_1 \times \dots \times F_n$ where Q_i is the state set of M_i and F_i is the accepting state set of M_i ($1 \leq i \leq n$). The components of the state of M are independent of each other, where the i th component simulates the change of state as dictated by M_i and the i th components of its neighbors' states. Whenever the i th component reaches a final state of M_i , it remains unchanged, while the other components continue to simulate the other M_j 's. Then $C(M)$ accepts Γ iff. each $C(M_i)$ accepts Γ iff. $\Gamma \in L_1 \cap \dots \cap L_n$. Also M is deterministic iff. every M_i is deterministic ($1 \leq i \leq n$). If each M_i is a recognizer of L_i , then M is a recognizer of $L_1 \cap \dots \cap L_n$ provided $C(M)$ rejects Γ as soon as one of the components (say the i th) of the state of M is in a rejecting state of M_i . //

Proposition 2: A finite union of (deterministic) d-graph languages (predicates) is a (deterministic) cellular d-graph language (predicate).

Proof: Define M' in the same way as M in Proposition 1 except that $S = (S_1, \dots, S_n)$ is an accepting state of M' iff. S_i is an accepting state of M_i for some i , $1 \leq i \leq n$. If the M_i 's are recognizers, $(S_1, \dots, S_n)$ is a rejecting state of M' iff. for every i , $1 \leq i \leq n$, S_i is a rejecting state of M_i . //

Proposition 3: Any singleton $\{\Gamma\}$ is a deterministic cellular d-graph predicate.

Proof: We can define a cellular d-graph recognizer M_Γ which discovers whether a d-graph is isomorphic to $U(\Gamma)$ [3], with the added requirement that M_Γ further checks the arc end numberings after isomorphism to $U(\Gamma)$ is confirmed. M_Γ accepts a d-graph only when all the arc end numberings are exactly the same as in Γ , and rejects all other d-graphs. Clearly M_Γ accepts only Γ . //

Proposition 4: Any finite set of d-graphs is a deterministic cellular d-graph predicate.

Proof: Propositions 2 and 3. //

The set complement of a cellular d-graph language L is the set of all d-graphs (having the same node label set as the d-graphs in L) that are not in L . Clearly cellular d-graph predicates are closed under complementation, but it is not known if the cellular d-graph languages are.

In graph theory, the complement of a graph G is a graph having the same nodes as G , and in which two nodes are joined

by an arc iff. they are not neighbors in G . However, the complement of a d -graph G is not necessarily a d -graph.

In fact, its nodes have degree $\leq (\text{the number of nodes in } G) - 1$, which is not bounded. Therefore it is not appropriate to consider complements of d -graphs.

3. Permutation

We next consider some "geometric" operations on cellular d-graph languages. For any d-graph Γ , a renumbering of Γ is a d-graph obtained by renumbering some (possibly none) or all of the arc ends of Γ ; thus it is a d-graph having the same underlying graph as Γ . For any cellular d-graph language L , the renumbering closure L_ρ of L is the set of all renumberings of the d-graphs in L . If the underlying graph of every d-graph in L is accepted strongly by a cellular d-graph acceptor, then $L_\rho = L$.

Proposition 5: The renumbering closure of a (deterministic) cellular d-graph predicate is a (deterministic) cellular d-graph predicate.

Proof: Let $C(M)$ recognize the cellular d-graph predicate P . Define a finite state recognizer A such that its states have a component which is a permutation of the numbers $1, 2, \dots, d$, and which is initially $(1, 2, \dots, d)$. This permutation vector is used to implicitly define a renumbering of the arc ends at a node. Specifically, the vector $(j_1, \dots, j_d)$ indicates that the i th arc end is now renumbered as j_i , $1 \leq i \leq d$. For any d-graph Γ , $A = (\Gamma, A, H)$ can first construct a depth-first spanning tree to get an ordering of the nodes [4]. Then it can change the permutation vector at each node systematically using the ordering of the nodes. Each change implicitly represents a new d-graph Γ_1 which is a renumbering of Γ . The states also have another component that stores the nodes' initial states. A restores the

initial states at the nodes of Γ and then simulates (Γ_1, M, H) . If it reaches an acceptance state of M then A accepts Γ since Γ is a renumbering of Γ_1 and $\Gamma_1 \in P$. Otherwise it reaches a rejecting state of M , and a change of the permutation vectors is made to give a new renumbering of Γ , where again A checks whether $C(M)$ accepts this new renumbering. If all the renumberings have been checked and the acceptance state was never reached, Γ is rejected. It is easy to see that A is deterministic if M is. //

It is not known whether the renumbering closure of a deterministic cellular d-graph language L is a deterministic cellular d-graph language. The construction in Proposition 5 does not work when L is not a predicate since for some renumbering Γ_1 of Γ , the cellular d-graph acceptor may not accept Γ_1 , and may not stop; thus it is not possible for A to test all the other renumberings which may be in L .

Proposition 6: The renumbering closure of a cellular d-graph language is a nondeterministic cellular d-graph language.

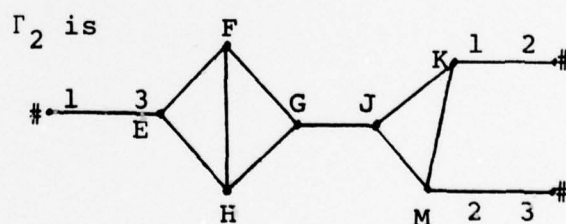
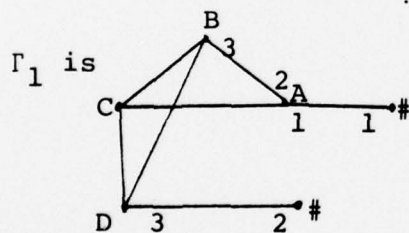
Proof: Define A' in the same way as A in Proposition 5 except that A' is nondeterministic, so that after getting a new numbering, $A' = (\Gamma, A', H)$ can start simulating (Γ_1, M, H) or can change to the next new numbering. Thus A' accepts Γ iff. one of the renumberings of Γ is in L iff. Γ is a renumbering of a d-graph in L . //

A renumbering is a permutation of the arc end numberings. We can also consider permutation of node labels. As in the proof of Proposition 5, we can order the nodes and then systematically test every permutation of their labels, so that this closure too is a deterministic cellular d-graph predicate. Another kind of permutation we can consider is permutation of the nodes. However, the resulting d-graph is just an automorphic image of the original one, and so is not distinguishable by a cellular d-graph automaton.

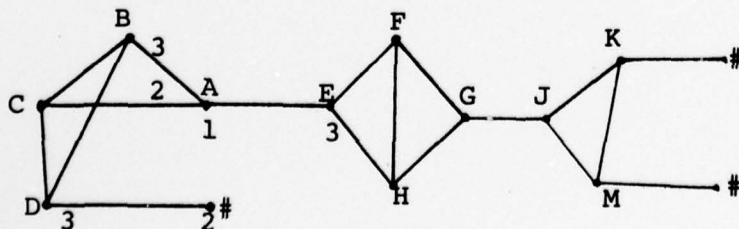
4. Concatenation and closure

Let there be given two d-graphs Γ_1, Γ_2 , where both Γ_1 and Γ_2 have a # node. A concatenation of Γ_1 and Γ_2 , denoted by $\Gamma_1 \cdot \Gamma_2$, is a d-graph obtained by performing the following operations: (1) delete the two # nodes n_1, n_2 and the arcs $(n_1, m_1), (n_2, m_2)$ joining n_1, n_2 to their neighbors m_1, m_2 in Γ_1, Γ_2 , respectively; (2) connect Γ_1 and Γ_2 with a new arc (m_1, m_2) ; and (3) assign the numberings of the arc ends $(m_1, n_1), (m_2, n_2)$ to the ends of this new arc at nodes m_1, m_2 . If one of the d-graphs does not have any # nodes, then concatenation is not defined.

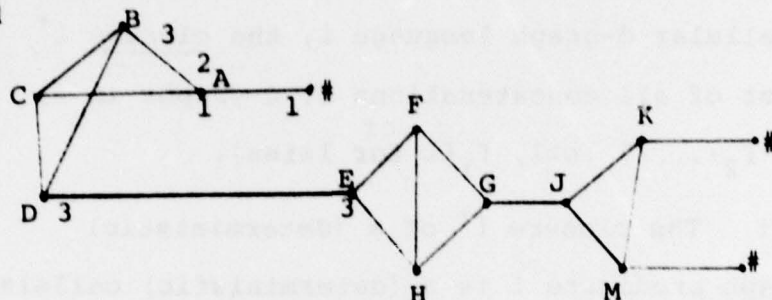
Two d-graphs may have many different concatenations, since each of them may have more than one # node, and there is a concatenation associated with each pair of # nodes of Γ_1 and Γ_2 . For example, if



then



and



are both concatenations of Γ_1 and Γ_2 .

It should be pointed out that if $\{\Gamma_1 \cdot \Gamma_2\}$ denotes the set of all concatenations of Γ_1 and Γ_2 then $\{\Gamma_1 \cdot \Gamma_2\} = \{\Gamma_2 \cdot \Gamma_1\}$. However, associativity does not hold, i.e. $\{(\Gamma_1 \cdot \Gamma_2) \cdot \Gamma_3\} \neq \{\Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_3)\}$, since

$$\begin{array}{c} \textcircled{\Gamma_3} - \textcircled{\Gamma_1} - \textcircled{\Gamma_2} \end{array} \in \{(\Gamma_1 \cdot \Gamma_2) \cdot \Gamma_3\} \text{ but } \notin \{\Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_3)\};$$

and if both Γ_1 and Γ_2 have only one # node, and Γ_3 has at least two # nodes then $\{(\Gamma_1 \cdot \Gamma_2) \cdot \Gamma_3\}$ is empty but $\{\Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_3)\}$ is not empty. We denote by $\Gamma_1 \cdot \Gamma_2 \cdot \dots \cdot \Gamma_n$ the set of d-graphs resulting from taking the concatenation of the n given d-graphs in any associative order. For example, $\{\Gamma_1 \cdot \Gamma_2 \cdot \Gamma_3\} = \{\Gamma_1 \cdot (\Gamma_2 \cdot \Gamma_3)\} \cup \{(\Gamma_1 \cdot \Gamma_2) \cdot \Gamma_3\}$.

The new arc joining Γ_1 and Γ_2 is always a bridge in $\Gamma_1 \cdot \Gamma_2$, and it is called a concatenative bridge. Not every bridge in $\Gamma_1 \cdot \Gamma_2$ is a concatenative bridge, since Γ_1 and Γ_2 may themselves have bridges.

For any cellular d-graph language L , the closure L^+ of L is the set of all concatenations of d-graphs in L , i.e. $L^+ = \{\Gamma_1 \cdot \Gamma_2 \cdot \dots \cdot \Gamma_n \mid n \geq 1, \Gamma_i \in L \text{ for } 1 \leq i \leq n\}$.

Proposition 7: The closure L^+ of a (deterministic) cellular d-graph predicate L is a (deterministic) cellular d-graph predicate.

Proof: In [4] it was shown that there exists a deterministic cellular d-graph automaton which identifies all the bridges of a d-graph and always terminates. Define a cellular d-graph recognizer M with a distinguished node D such that for any d-graph Γ , it identifies all the bridges, and also gives an ordering to the bridges. Each bridge may or may not be a concatenative bridge. M can systematically consider different subsets of the set of bridges as concatenative bridges. For each subset of bridges of size k , $0 \leq k \leq t$ (t = the number of bridges in Γ), if each bridge end-node treats the neighbor at the other end of the bridge as a # node, then Γ is implicitly decomposed into $k+1$ components. Let D , together with the bridge end-node further away from D in a breadth-first spanning tree, be the distinguished nodes of the components that they lie in. (Note that each component has only one distinguished node.) M can then test to see whether each component belongs to L . As soon as either component is rejected, M restores the initial states

of the nodes and proceeds to check another subset of bridges. M accepts Γ if it succeeds in finding a set of bridges such that all the components are in L . If all 2^t sets of bridges are checked without success, M rejects Γ . Clearly M is deterministic if L is. //

Proposition 8: The closure L^+ of a cellular d-graph language L is a nondeterministic cellular d-graph language.

Proof: Define a cellular d-graph acceptor A similar to M in Proposition 7 except that each time a subset of bridges is identified, A can nondeterministically choose to check if the components are in L , or can go on to test another set of bridges. //

The concatenation of the cellular d-graph languages $L_1, L_2, \dots, L_k$ is the set of d-graphs $L_1 \cdot L_2 \cdot \dots \cdot L_k = \{\Gamma_1 \cdot \Gamma_2 \cdot \dots \cdot \Gamma_k \mid \Gamma_i \in L_i, 1 \leq i \leq k\}$.

Proposition 9: A concatenation of (deterministic) cellular d-graph predicates is a (deterministic) cellular d-graph predicate.

Proof: Let $L_1, \dots, L_k$ be cellular d-graph predicates. The proof is analogous to that of Proposition 7, except that now M decomposes Γ into exactly k components, and for each subset of $k-1$ bridges, M systematically tests the $k!$ ways that the components can be in $L_1, \dots, L_k$, since the ordering of the bridges gives an ordering of the components. //

Proposition 10: A concatenation of cellular d-graph languages is a nondeterministic cellular d-graph language.

Proof: The proof of Proposition 9 can be modified the same way as in Proposition 8. //

We do not know whether closures or concatenations of deterministic cellular d-graph languages, which are not predicates, are deterministic cellular d-graph languages.

5. Line graphs

The line graph of a graph G , $L(G)$, is the graph obtained by creating a node for each arc of G and joining together those nodes corresponding to arcs that are adjacent (i.e., have a common endpoint) in G . If the degree of G is bounded by d , the degree of $L(G)$ is bounded by $2(d-1)$. If G_1 and G_2 are isomorphic, then obviously $L(G_1)$ and $L(G_2)$ are. The following two theorems give a characterization of line graphs.

Theorem (Whitney): Let G and G' be connected graphs with isomorphic line graphs. Then G and G' are isomorphic unless one is C_3 and the other is $K_{1,3}$ where C_3 is the graph  and $K_{1,3}$ is .

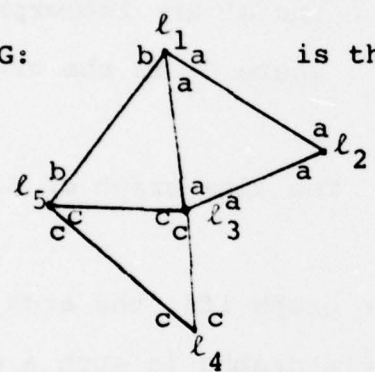
Proof: See [5] p. 72 or [6]. The line graph of $K_{1,3}$ or of C_3 is C_3 . //

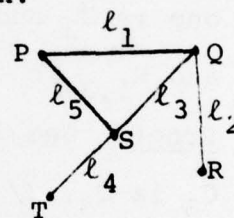
Theorem (Krausz): H is a line graph iff. the arcs of H can be partitioned into complete graphs in such a way that no point lies in more than two of the subgraphs.

Proof: See [5] p. 74 or [7]. //

Intuitively, the arcs of a line graph $H = L(G)$ at each node can be divided into two classes, one associated with each end of the arc of G which n represents. All the arcs incident upon the same node of G form a complete subgraph of H . Given the partition of arcs at each node of H as in Krausz's theorem, G can be readily constructed. By Whitney's theorem, for a line graph which is not C_3 , there is only one way up to

isomorphism to partition the arcs at the nodes so as to satisfy the criteria of Krausz's theorem. There may be more than one way to partition the arcs at a node into two classes which form complete subgraphs with the nodes at the other end of the arcs. However, if the partition at a node is not correct and we continue to partition the arcs of the other nodes, then we will reach a node whose arcs cannot be partitioned so as to satisfy Krausz's criteria because otherwise Whitney's theorem will be violated. For example,

the graph G:  is the line graph of H:



The triangle (complete graph of three nodes) in G defined by l_1, l_2, l_3 is associated with node Q of H because l_1, l_2, l_3 are the three arcs incident upon Q. The complete graph in G associated with node R of H contains only one node l_2 .

The correct partitions of the arcs at the nodes of G are indicated in the above figure; the arcs with the same labels (a, b, or c) belong to the same class. Suppose that in the partitioning process, the arcs at node l_1 were erroneously divided into the two classes $\{(l_1, l_3), (l_1, l_5)\}$ and

$\{(\ell_1, \ell_2)\}$. This forces the partition at node ℓ_2 to be $\{(\ell_2, \ell_1)\}, \{(\ell_2, \ell_3)\}$. Therefore at node ℓ_3 , $\{(\ell_3, \ell_1), (\ell_3, \ell_5)\}$ is one class of arcs, $\{(\ell_3, \ell_2)\}$ is another class, and the arc (ℓ_3, ℓ_4) belongs to neither class. Hence the mistake made at node ℓ_1 is discovered.

A line graph of a labelled graph γ , denoted by $L(\gamma)$, is a labelled graph such that:

- (1) the node label set of $L(\Gamma)$ is $S \times S$, where S is the set of node labels of Γ ;
- (2) if x represents arc (m, n) of Γ and the labels of m, n are A, B , then node x of $L(\Gamma)$ has label either (A, B) or (B, A) .

A line graph of a d-graph Γ , denoted by $L(\Gamma)$, is a $2(d-1)$ -graph such that the underlying graph of $L(\Gamma)$ is a line graph of $U(\Gamma)$, the underlying graph of Γ . A d-graph Γ has many line graphs, since a node of $L(\Gamma)$ may be labelled (A, B) or (B, A) and there are many ways to number the arcs. The line graph set $L(L)$ of a d-graph language L is the set of all line graphs of d-graphs in L . In considering line graphs, the main interest is in the structure of the graph; therefore we have often ignored the labels of the non-# nodes, or assumed that all the non-# nodes have the same label, which is the same as if the non-# nodes were not labelled. We will call such d-graphs unlabelled d-graphs.

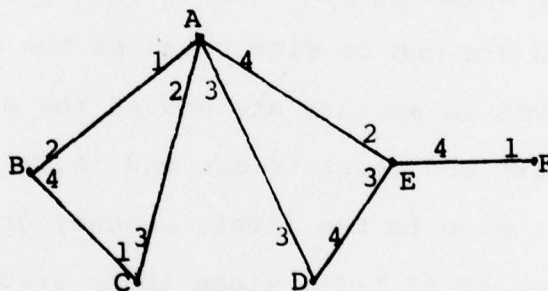
Proposition 11: There exists a deterministic cellular d-graph recognizer M with a distinguished node that accepts all (unlabelled) d-graphs whose underlying graphs are line graphs. Moreover, when M accepts, the arcs at each node are partitioned into two classes of which one may be empty and the arcs in each class belong to a complete subgraph.

Proof: M first verifies whether the underlying graph is C_3 and accepts the d-graph if it is. Otherwise, M constructs a depth-first spanning tree and an ordering of the nodes as in [4]. Each node's state has a classification component $(a_1, \dots, a_d)$ where $a_i \in \{\#, b, r, N\}$. Here $a_i = \#$ if the i th neighbor is a $\#$ node; otherwise initially $a_i = N$ and it is changed to b or r later to indicate which class the i th arc belongs to. M can systematically (according to the ordering of the nodes) divide the non- $\#$ arcs at each node into two classes (one may be empty) in all possible ways and test whether they form complete subgraphs as in Krausz's theorem. The test for complete subgraphs is the same as the test for complete graphs in [2], except that now the nodes which are not at the other ends of the arcs in the same class are ignored. If any partition of the arcs is successful then M accepts the d-graph, since its underlying graph is a line graph; and the classification components at each node tell how the arcs at each node are partitioned. //

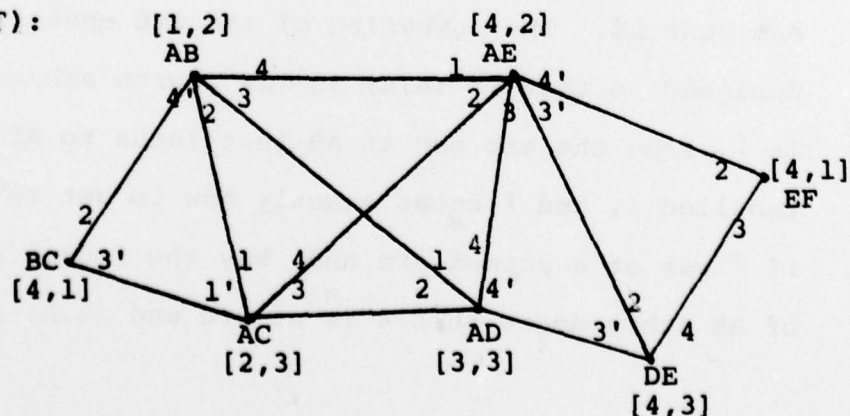
If the d-graph is labelled, and the nodes have labels in $S \times S$ for some label set S , then the partitioning of the arcs at the nodes with labels (ℓ_1, ℓ_2) where $\ell_1 \neq \ell_2$ is much simpler, since the arcs leading to nodes having ℓ_1 in their labels form one class and those leading to arcs having ℓ_2 in their labels form another class. If a neighbor does not have ℓ_1 or ℓ_2 in its label, then the d-graph is not a line graph.

We will now show that labelling the arc ends of the line graph $L(\Gamma)$ in a special way will allow a sequential $(2d-2)$ -graph automaton Λ on $L(\Gamma)$ to simulate a sequential d-graph automaton F on Γ . First let us look at an example:

Γ :



$L(\Gamma)$:



The arc ends of $L(\Gamma)$ at a node x are numbered so that the arcs associated with one end of the arc of Γ that x represents are assigned primed numbers, while those associated with the other end are assigned unprimed numbers.

Suppose A is located at some arc end of Γ , say arc end (A,B) near node A . Then F should be on some arc end at node AB of $L(\Gamma)$, and this arc should be associated with node A , i.e. should be part of the subgraph induced by node A . Therefore F may be on the second, third, or fourth arc end at node AB of $L(\Gamma)$. There are two kinds of moves A can make:

- (1) A moves to the other end of the same arc. Then F simply needs to move from an unprimed arc end to a primed arc end or vice versa at the same node.
- (2) A moves to another arc end at the same node, say from arc end (A,B) to arc end (A,E) at node A . Then F must move to the first, second, or third arc end of node AE of $L(\Gamma)$, since these arcs are induced by node A . This means that F first moves to the arc end at AB , which would lead to AE , and then moves to the end near AE . The numbering of the arc ends is designed so that if (A,E) is the fourth arc end at A in Γ , then the arc end at AB that leads to AE is labelled 4, and F knows exactly how to get to node AE . If F was at a primed arc end, say the fourth arc end of AE (this means that A is at arc end (E,A) at node E),

then to simulate A moving to the third arc end at E, F moves to a primed arc end, the third arc end of AE, and then moves toward DE.

This numbering of the arc ends of $L(\Gamma)$ can always be done because in the complete subgraph S of $L(\Gamma)$ induced by a node n of Γ , the nodes of $L(\Gamma)$ represent the arcs of Γ near node n . We can give each node of S the number of the arc end that it represents. These numbers are all distinct. Thus all the arc ends in S leading toward the node with number i are all numbered with i . The arc ends at the nodes of S that do not belong to S will have primed numbers in order to avoid confusion. Of course, these primed and unprimed numbers are not the actual arc end numbers of $L(\Gamma)$, but they can be defined implicitly. Moreover, one of the numbers at the ends of an arc may be primed while the other is unprimed; as an example, see the arc joining AB and BC in $L(\Gamma)$.

Proposition 12: Let L be a (deterministic) cellular d-graph language (or predicate) such that for any d-graph $\Pi \in L$, the renumbering of Π is also in L . Then $L(L)$ is a (deterministic) cellular d-graph language (predicate).

Proof: Let $C(M)$ accept L . Let $\hat{M} = (\Gamma, M, H)$ be a cellular $(2d-2)$ -graph acceptor such that it first determines if Γ is a line graph by imitating the recognizer in Proposition 11. Suppose Γ is a line graph; then each node's state has a

classification vector. $\hat{M}$ makes sure that each class has at most $d-1$ elements, since each complete subgraph is induced by the arcs at a node of a d -graph. Now $\hat{M}$ proceeds to implicitly assign primed and unprimed numbers to the arc ends as follows: for each complete subgraph S , $\hat{M}$ gives each node of S a distinct number in $Z_d = \{1, \dots, d\}$. Hence each node x of Γ has a pair of numbers, of which the first one corresponds to its number in the subgraph specified by the b 's in its classification vector, while the second one corresponds to that of the r 's. Now $\hat{M}$ rewrites the classification vector $(a_1, \dots, a_{2d-2})$ of a node n into $(m_1, \dots, m_{2d-2})$ so that when the numbers of n 's i th neighbor are (c_1, c_2) , if $a_i = b$ then $m_i = c_1$, and if $a_i = r$ then $m_i = c_2'$. After the non-# a_i 's have changed, the other m_j 's are assigned numbers so that $d-1$ of the m_j 's are distinct and in Z_d , the other $d-1$ are distinct and in $Z_d' = \{1', \dots, d'\}$, and $m_j \neq n_1, m_j \neq n_2'$ for any $1 \leq j \leq d-2$ where (n_1, n_2) is the pair of numbers of node n . The classification vectors implicitly give a numbering of a d -graph Π such that $\Gamma = L(\Pi)$.

Let A be a sequential d -graph acceptor that simulates the cellular d -graph acceptor $M = (\Pi, M, H)$ on Π [1]. Now define a sequential $(2d-2)$ -graph acceptor $\hat{A}$ such that:

- (1) If A moves to the other end of the same arc, then $\hat{A}$ moves from an unprimed arc end to a primed arc end or vice versa at the same node.

- (2) Suppose A moves to the i th arc end at the same node, and $\hat{A}$ is on an arc end with an unprimed number. If i is not in the node's classification vector then $\hat{A}$ need not move; otherwise $\hat{A}$ moves to the arc end possessing the number i and then moves to the other end of that arc.

From the discussion preceding the proposition, $\hat{A}$ on this assignment of Γ simulates A on Π . Therefore $\hat{A}$ accepts this assignment of Γ iff. A accepts Π , i.e. $\Pi \in L$.

$\hat{M}$ can simulate $\hat{A}$ on Γ . $\hat{M}$ accepts Γ iff. $\hat{A}$ accepts, i.e. $\Pi \in L$. Since Π is a d -graph such that Γ is its line graph, and any renumbering of Π is also in L , $\hat{M}$ accepts $L(L)$. Note that if it is not known whether a renumbering of Π is in L , then we cannot conclude that $\Gamma \in L(L)$ even though $\Pi \in L$, since there may be a renumbering of Π in L and Γ is its line graph.

Clearly, $\hat{M}$ is deterministic if M is and $\hat{M}$ is a recognizer if M is. //

Proposition 13: The line graph set $L(L)$ of a (deterministic) cellular d -graph predicate L is a (deterministic) cellular $(2d-2)$ -graph predicate.

Proof: For any $(2d-2)$ graph Γ , each node belongs to at most two complete subgraphs. The ordering of the nodes of Γ induces an ordering of the complete subgraphs of Γ . Therefore M of Proposition 12 can be modified to systematically assign primed and unprimed numbers to the nodes and thus

implicitly define different d-graphs with the same underlying graph. When $\hat{A}$ accepts, $\hat{M}$ accepts. If $\hat{A}$ rejects, $\hat{M}$ tries a new assignment of numbers and hence a different numbering of the d-graph. M rejects when all possible numberings fail. //

Proposition 14: The line graph set $L(L)$ of a cellular d-graph language is a nondeterministic cellular $(2d-2)$ -graph language.

Proof: Similar to Proposition 13 except that when all the nodes receive a new pair of number assignments, the cellular $(2d-2)$ -graph acceptor M' may nondeterministically proceed as $\hat{M}$ or may start giving the nodes another number assignment. //

6. Concluding remarks

Cellular d-graph languages have been shown to be closed under set-theoretic operations such as finite union and intersection, and geometric operations such as permutation (of arc end numbering and node labels), concatenation, and formation of line graphs. Under the set-theoretic operations, determinism is always preserved. However, under the geometric operation, determinism is known to be preserved only when the languages are also predicates.

Many of the common operations on graphs such as join, product and composition were not considered in this paper, because there is no bound on the degrees of the resulting graphs; their degrees depend on the numbers of nodes in the given graphs. The (geometric) union of two connected graphs gives a graph that is not connected.

Another geometric operation which preserves connectivity and boundedness of degree on d-graphs is simple (or elementary) contraction. A simple contraction of a graph G is obtained by identifying two adjacent nodes u, v , i.e., by the removal of u and v and the addition of a new node w adjacent to those points to which u or v was adjacent. A simple contraction of a degree d graph has degree $\leq 2d$. A simple contraction of a d-graph Γ is a $2d$ -graph whose underlying graph is a simple contraction of the underlying graph of Γ . A simple contraction of a cellular d-graph language L is a cellular $2d$ -graph

language, since if there is a node with more than d non-# neighbors then we can partition its arcs into two sets of d arcs and treat it as two nodes to test whether the d -graph is in L . If all nodes have $\leq d$ non-# neighbors, then one has to systematically test every partition of the arcs at every node into two sets of d arcs each. The operation of contraction, meaning a sequence of elementary contradictions, can produce graphs of arbitrarily high degree.

References

1. A. Wu, Cellular graph acceptors, TR-599, Computer Science Center, University of Maryland, College Park, MD, November 1977.
2. A. Wu, Cellular graph acceptors, 2, TR-621, Computer Science Center, University of Maryland, College Park, MD, December 1977.
3. A. Wu, Cellular graph acceptors, 3, TR-648, Computer Science Center, University of Maryland, College Park, MD, April 1978.
4. A. Wu, Cellular graph acceptors, 4, TR-667, Computer Science Center, University of Maryland, College Park, MD, June 1978.
5. F. Harary, Graph Theory, Addison-Wesley, Reading, Massachusetts, 1969.
6. H. Whitney, Congruent graphs and the connectivity of graphs, Amer. J. Math. 54, 1932, pp. 150-168.
7. J. Krausz, Démonstration nouvelle d'un théorème de Whitney sur les réseaux, Mat. Fiz. Lapok 50, 1943, pp. 75-89.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER AFOSR-TR-78-1372	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) CELLULAR GRAPH ACCEPTORS, 5: CLOSURE PROPERTIES OF CELLULAR d-GRAPH LANGUAGES		5. TYPE OF REPORT & PERIOD COVERED Interim Rept.	
7. AUTHOR(s) Angela Wu		6. PERFORMING ORG. REPORT NUMBER	
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Maryland Computer Science Center College Park, Maryland 20742		8. CONTRACT OR GRANT NUMBER(s) AFOSR-77-3271	
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, DC 20332		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2 A2	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		13. REPORT DATE July 1978	
		12. NUMBER OF PAGES 29	
		15. SECURITY CLASS. (of this report) UNCLASSIFIED	
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited. (12) 31 p. (14) TR-681			
17. DISTRIBUTION STATEMENT (of abstract entered in Block 20, if different from Report)			
18. SUPPLEMENTARY NOTES			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Cellular Automata Formal Languages Closure Properties Graphs			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Cellular d-graph languages are shown to be closed under set theoretic operations, including finite union and intersection; and under geometric operations, including permutation of arc end numbering, concatenation, closure, and formation of line graphs. Determinism is preserved under the set-theoretic operations; but under the geometric operations, determinism is known to be preserved only when the languages are also predicates.			