PATTERN SYNTHESIS AND PERTURBATION IN TESSELLATION AUTOMATA

by

S. M. Walters

Dissertation submitted to the Graduate Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

APPROVED:

Richard A. Thompson
R. A. Thompson, Chairman

F. G. Gray

D. P. Roselle

E. A. Manus

H. F. VanLandingham

December 1977

Blacksburg, Virginia

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to each and every member of his doctoral committee. Special recognition is given to his committee chairman, Professor Richard A. Thompson, for his patience, guidance and stimulation, and Professor F. Gail Gray for his invaluable assistance during the final stages of research.

This research was sponsored by the National Science Foundation under grants DCR75-06543 and MCS75-06543 A01 and the author gratefully acknowledges this support. The financial support from the Electrical Engineering Department of Virginia Polytechnic Institute and State University is also thankfully acknowledged. Also acknowledged is Cheryl West for a superb typing of a difficult manuscript.

A special note of tribute is due to my patient and understanding wife, Mitsu, for her continuous encouragement, and to our firstborn child, Michelle Elizabeth, to whom I dedicate this dissertation.

TABLE OF CONTENTS

		Page
I.	INTRODUCTION TO CELLULAR AUTOMATA.	1
II.	THE TESSELLATION AUTOMATON	5
III.	A CELLULAR COMPUTER.	14
IV.	TURING MACHINES AND TESSELLATION AUTOMATA.	20
	4.1 Introduction.	20
	4.2 Turing Machines	20
	4.3 Behavioral Equivalence.	23
	4.4 Stability of Tessellation Automata.	32
	4.5 Discussion.	38
V.	PATTERNS	39
	5.1 Introduction.	39
	5.2 Strings	39
	5.3 Extended Dimension.	45
	5.4 Irregular Patterns.	54
	5.5 An Example.	57
VI.	GROWTH	62
	6.1 Introduction.	62
	6.2 One-dimensional Growth.	65
	6.3 Multi-dimensional Growth.	68
	6.4 Irregular Neighborhoods and Configurations.	71
	6.5 Discussion.	76
VII.	SYNTHESIS.	78
	7.1 Introduction.	78

Page

7.2 Synthesis in One-Dimensional Automata 84

7.3 Synthesis in Multi-dimensional Automata 93

7.4 Discussion. 104

VIII. PERTURBATION 107

8.1 Introduction. 107

8.2 Restoration of Equilibrium Configurations 108

8.3 Self-Diagnosis of Single Perturbations. 114

8.4 Reduction in Alphabet 135

8.5 Discussion. 144

IX. SUMMARY. 145

BIBLIOGRAPHY 154

VITA 160

ABSTRACT

I. INTRODUCTION TO CELLULAR AUTOMATA

With the advent of integrated circuit technology, the notion of constructing systems from cellular "building blocks" has become very attractive. If the cells are interconnected in a uniform fashion, fabrication of printed circuit boards to construct the system also becomes very straightforward. In spite of an early start, theoretical advances in this area lag far behind technology. Some special purpose applications which lend themselves to a cellular decomposition have been implemented, such as array processors which compute matrix addition, multiplication, fast fourier transforms and other functions which are of a highly "parallel" nature. The binary adder of Figure 1.1 enjoys widespread use as a favorite mechanism used to introduce students to cellular realizations. Random access memories used in most processors today are of a highly cellular nature. These applications are of a very practical nature and, certainly, there must exist other equally useful applications which remain undiscovered.

Hennie [10], [11], [12], has investigated a class of cellular automata known as iterative arrays. These are classed as unilateral or bilateral according to the interconnection of the cells and sequential or combinational as to cell type. Examples of these arrays are shown in Figure 1.2. Hennie solved a number of difficult theoretical questions concerning iterative arrays and demonstrated a synthesis technique for specifying the cell's state table to perform some given computation.

Unger [24] demonstrated the possibility of doing pattern recognition using iterative arrays. In doing so, pattern features such as right or left concavity, circularity, etc., are extracted. This method

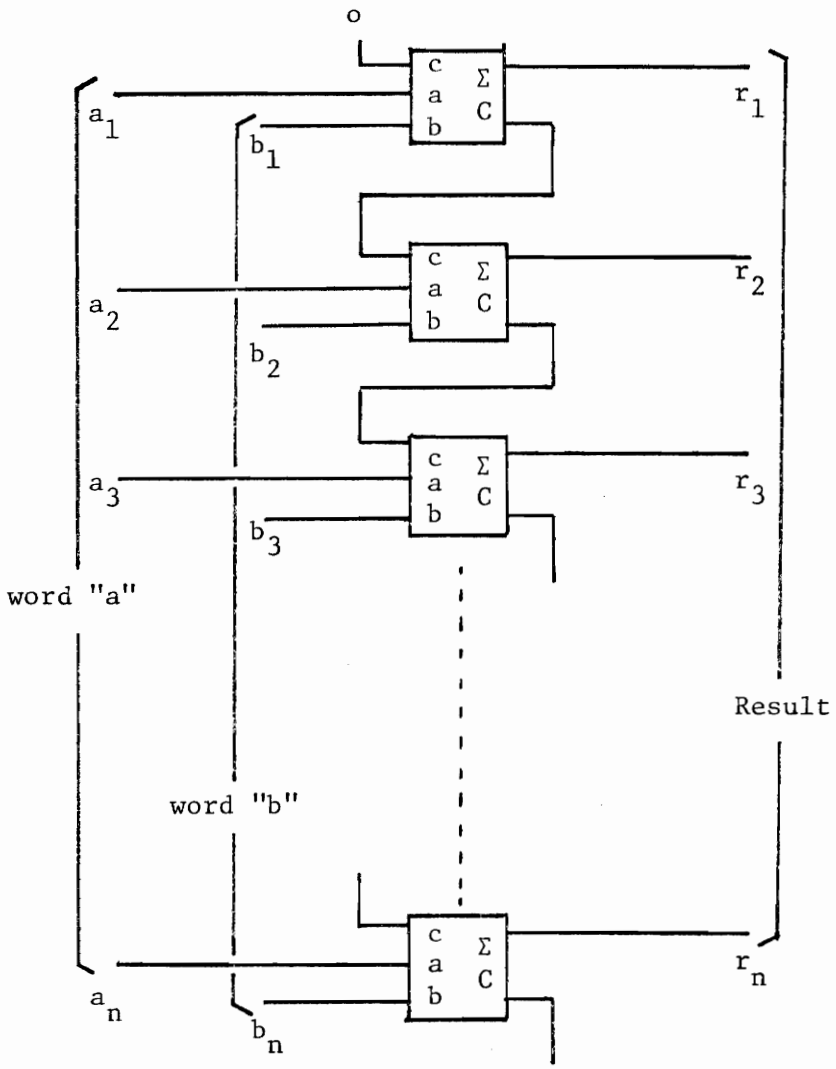


Fig. 1.1 A Parallel Adder

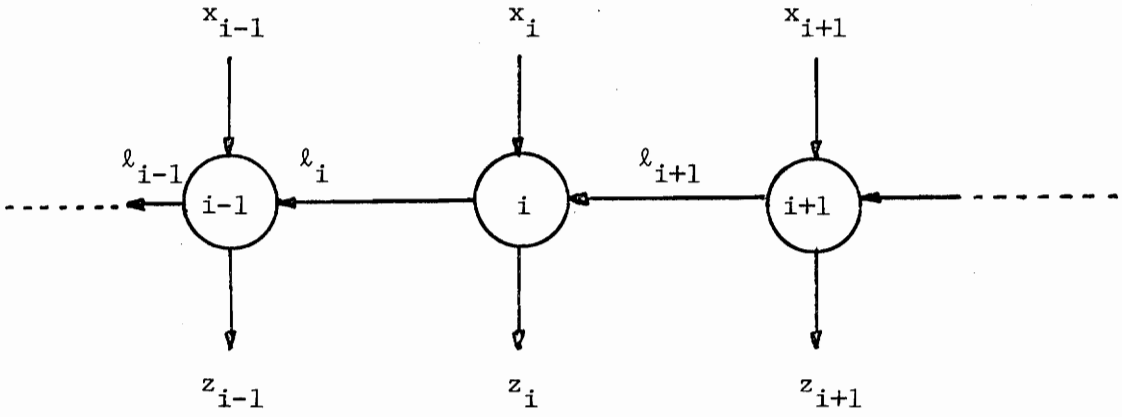


Fig. 1.2a A Unilateral Cellular Array

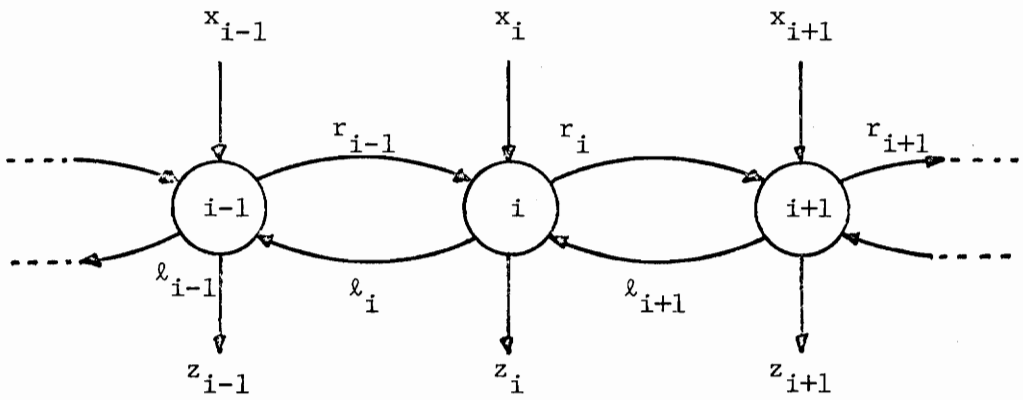


Fig. 1.2b A Bilateral Cellular Array

Fig. 1.2 Two Examples of Cellular Arrays

of finding pattern features has some very real advantages but the problem of determining the pattern from the features remains unsolved.

Holland [13] has described a general purpose computer which is cellular and demonstrated its equivalence to Turing machines. Unfortunately, no technique for programming it has been found. For this reason, no implementation has been attempted.

More recent efforts have involved the creation of arrays where each cell is a more "standard" computer. An example of this approach is the ILLIAC IV computer. In doing so, each cell is of a very powerful nature and, although modularity is retained, the modules themselves are very costly. The introduction of the microprocessor with its associated low cost may make this approach quite effective, but it remains largely uninvestigated.

The main point to be made is that a great many unsolved problems stand between wishes and reality. Furthermore, the advantages of a cellular computer are unclear to technologists. After all, general purpose computers already exist and perform admirably. Technology is constantly making them bigger and faster. Programming techniques are well known and the recent appearance of home computer systems has made "BASIC" and "FORTRAN" household words. Speed, modularity and ease of construction are the most common justifications offered, but the fault tolerant aspect of cellular mechanisms is perhaps the best. Imagine a computer which could continue doing accurate computation with defective components scattered throughout its structure. It is the purpose of this dissertation to demonstrate, in part, such a possibility.

II. THE TESSELLATION AUTOMATON

The formal notion of a tessellation automaton (TA) is a four-tuple,

$$TA = (A, E^d, X, I)$$

where:

(1) A is a finite non-empty set called the state alphabet of TA. The set, A, represents the set of states that can be assumed by any machine in the array of machines being modeled.

(2) d is a positive integer greater than zero called the tessellation dimension and E^d is the set of all d-tuples of integers called the tessellation space. The elements of E^d are locations within the space and there is a machine at each location.

(3) X is an n-tuple of distinct d-tuples of integers and is called the neighborhood index for the TA. It is used to define the uniform interconnection pattern among the machines in the array.

(4) Any mapping $c: E^d \rightarrow A$ is called a (global) configuration and C denotes the set of all such mappings. The image of $i \in E^d$ under $c \in C$ is written as c(i) and is referred to as the content of cell i in configuration c. If $X = (\xi_1, \ldots, \xi_n)$ and $i \in E^d$, then $N(X,i) = (i + \xi_1, \ldots, i + \xi_n)$ is an n-tuple of d-tuples called the neighborhood of cell i. Then c(N(X,i)) is an n-tuple called the neighborhood configuration of cell i where $c(N(X,i)) = (c(i + \xi_1), c(i + \xi_2), \ldots, c(i + \xi_n))$. This is simply the content of the neighbors of cell i. Notice that $\forall c$, i,

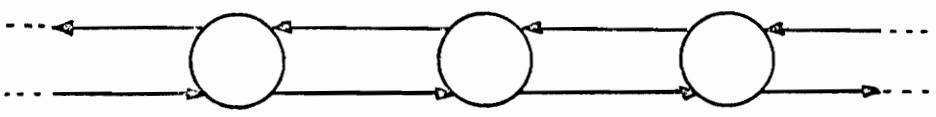
$c(N(X,i)) \in A^n$. Let any mapping $\sigma: A^n \rightarrow A$ be called a local transformation and let L be the set of all such mappings. The next state for each cell, i, is denoted as $c'(i)$ and is found using σ where $c'(i) = \sigma(c(N(X, i)))$. If σ is applied to every cell, the next (global) configuration, c', is found. Define a mapping, $\tau: C \rightarrow C$ where $\forall c \in C$, $\tau(c) = c'$ if and only if $\forall i$, $\sigma(c(N(X,i))) = c'(i)$. Then τ is called the global transformation and finds the next (global) configuration, $\tau(c) = c'$. Notice that for a given σ there is a corresponding τ. For that reason, we say that σ implies τ ($\sigma \Rightarrow \tau$). Let T be the set of all global transformations where $\forall \sigma \in L$, $\sigma \Rightarrow \tau$, $\tau \in T$. Notice that there is a bijection $\delta: L \rightarrow T$. Finally, any non-empty subset of T is referred to as I. Each of the elements $\tau \in I$ has a corresponding σ where $\tau = \delta(\sigma)$ and $\sigma = \delta^{-1}(\tau)$. The subset of L which determined I is referred to as $\delta^{-1}(I)$ and is the set of local transformations which are actually "wired in" for the cells in the automaton.

This formalization is due to Yamada and Amoroso [28].

To demonstrate, a tessellation automaton is defined.

$$TA = (\{0,1\}, E^1, (-1,1), I)$$

If the array is constructed, it will appear as:



This is due to the tessellation space, E^1, and X, the neighborhood index. Since $|X| = n = 2$, then $\sigma: A^2 \rightarrow A$. All possible local transformations can be listed. These make up the set, L, of all local transformations.

c(N(X,i))	$\sigma_0\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6\sigma_7\sigma_8\sigma_9\sigma_{10}\sigma_{11}\sigma_{12}\sigma_{13}\sigma_{14}\sigma_{15}$
00	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
01	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
10	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
11	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

↑

Since the space E^2 is infinite, then C, the set of all configurations, is infinite. Since τ: C → C, τ cannot be written in its entirety. We can find $\tau(c)$ for a few configurations using some σ. For instance, if σ_6 is selected,

c	$\tau_6(c)$
$\overline{0}$ 0 0 1 0 0 $\overline{0}$	$\overline{0}$ 0 1 0 1 0 $\overline{0}$
$\overline{0}$ 0 1 1 0 0 $\overline{0}$	$\overline{0}$ 1 1 1 1 0 $\overline{0}$
$\overline{0}$ 0 1 1 1 0 $\overline{0}$	$\overline{0}$ 1 1 0 1 1 $\overline{0}$
etc.	etc.

For each local transformation, σ_i, there is a corresponding global transformation, τ_i and T is the set of all τ_i. Then a choice for I is I = $\{\tau_1, \tau_6, \tau_{15}\}$. Any subset of T is a valid choice for I.

Some special considerations follow from the definition. Let ξ_o be the d-tuple of all zeroes. If $\xi_o \in X$, then each cell is in its own neighborhood. Then, the next state for each cell depends on its neighbor's states and its own state. Within this dissertation, frequent reference to ξ_o will be made.

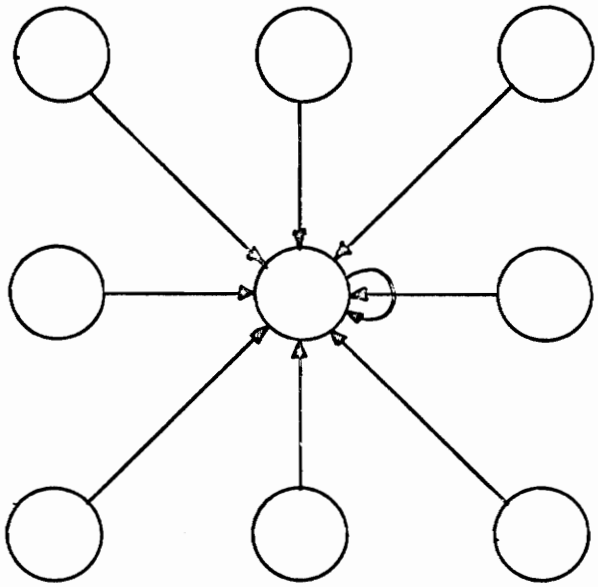
A d-dimensional neighborhood index of the form $X = (\xi_1, \ldots, \xi_n)$ where each component is an element of the set $\{(z_1, \ldots, z_d) | z_j \in$

$\{-1,0,1\}$, $1 \leq j \leq d\}$, is called a partial Moore neighborhood index. If all 3^d components are in, X is called a Moore neighborhood index and is denoted X_M. The two-dimensional Moore neighborhood index is shown in Figure 2.1.

By a d-dimensional von Neumann neighborhood index, we mean the (2d + 1)-tuple $X = (\xi_o, \ldots, \xi_{2d})$ where one element is ξ_o and the rest are of the form $(0,\ldots, 0, z, 0, \ldots, 0)$ where $z \in \{-1,1\}$. Then each element in X is either +1 or -1 in a single component except for ξ_o. Figure 2.2 shows the two-dimensional von Neumann neighborhood index. The formalization of these two neighborhood indexes is due to Yamada and Amoroso [30].

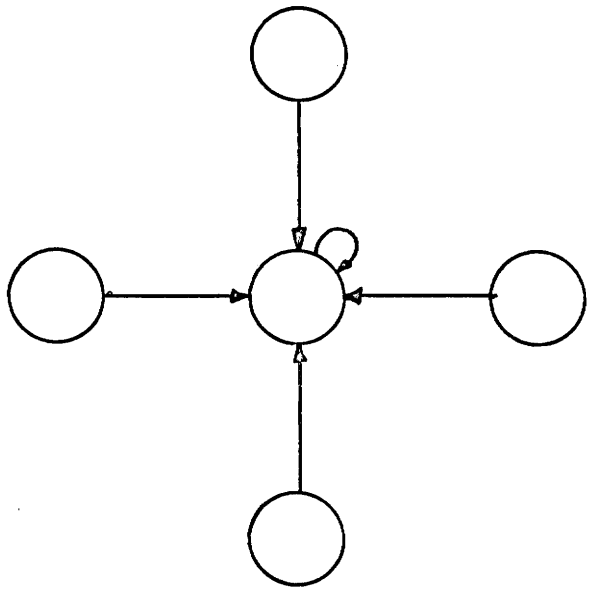
The case where I contains a single global transformation, τ, will be of great interest in this dissertation. For the given global transformation, there is a single local transformation, $\sigma = \delta^{-1}(\tau)$. For this case, the automaton $TA = (A, E^d, X, \sigma)$ is defined and will be referred to as an autonomous tessellation automaton.

The tessellation automaton was first studied by von Neumann [25] who investigated self-reproduction in two-dimensional automata using the neighborhood index which is now named for him. (This work was edited and published in 1966 by A. W. Burks after von Neumann's death.) Von Neumann described an automaton made up of 29 state cells which was capable of constructing a copy of itself. The copy was then released and both machines could then create further copies of themselves. It was later shown by Thatcher [21] that von Neumann's model contained a universal Turing machine which is self-describing. E. F. Moore [16] was



X = ((-1,1),(0,1),(1,1),(-1,0),(0,0),(1,0),(-1,-1),(0,-1),(1,-1))

Fig. 2.1 The Two-dimensional Moore Neighborhood Index



X = ((0,0), (0,1), (1,0), (0,-1), (-1,0))

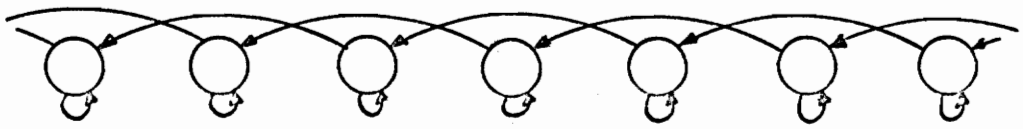
Fig. 2.2 The Two-dimensional von Neumann Neighborhood Index

the first person known to call these automata "tessellations". He, too, was interested in self-reproduction and raised an interesting question which has come to be known as the "Garden-ofEden" problem. In doing so, he placed von Neumann's questions concerning self-reproduction in a more abstract framework. The "Garden-of-Eden" question deals with the possibility that some configurations may never arise in the automaton unless they are placed in it as initial configurations. If no such configurations exist, the automaton is called complete. Otherwise, it is called incomplete.

Yamada and Amoroso [28] set forth the notational description which we have stated previously and produced results concerning laminated tessellation automata. An automaton which is laminated has two or more independent tessellation automata sharing the same tessellation space. This can occur if certain conditions on the neighborhood index are not met. Several examples of laminated automata are shown in Figure 2.3.

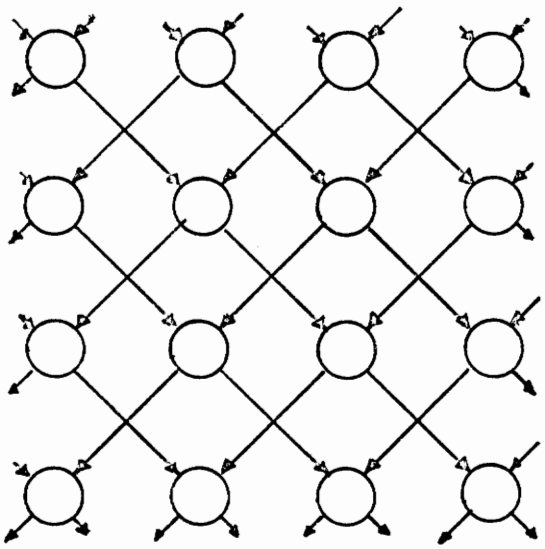
They later introduced results [9] concerning completeness which demonstrate that one-dimensional automata with a binary alphabet are complete if the neighborhood index is scope-4 or more (the notion of "scope" is formalized in Definition 6.2.1 of this dissertation) and that scope-2 automata are incomplete. Results for certain two-dimensional automata are also presented. This work was extended to non-binary alphabets by Amoroso and Cooper [1].

Yamada and Amoroso also considered [30] the question of equivalence in tessellation automata. In this report, both structural and behavioral equivalences are defined and studied. In doing so, equivalence due



X = (0,2)

X = (0,3)



X = ((1,1), (-1,1))

Fig. 2.3 Some Laminated Tessellation Automata

to coordinate transformation and neighborhood shifts are found. Amoroso and Guilfoyle [3] continued this effort and found equivalent automata in two dimensions which have three elements in the neighborhood index. Amoroso and Epstein [4] considered the possibility of equivalent automata with permuted neighborhoods. Studies concerning the injectivity of the global transformation have been produced by Patt [18], and Amoroso and Patt [2]. These studies consider the existence of Garden-of-Eden configurations since any automaton having an injective global transformation will have them.

As is seen, most of these studies relate to either self-reproduction or evolutionary studies. It is our purpose to employ the tessellation automaton as a controlling device rather than a biological model. Of closest relevance to our purposes are the results pertaining to equivalence. Even these results are of little value with respect to our particular interest.

III. A CELLULAR COMPUTER

It is the purpose of this chapter to present the cellular structure to which the results of this dissertation may be applied. The structure is shown in Figure 3.1. As can be seen, it consists of two cellular arrays which are interconnected. The two arrays are called the "control" hyperplane and the "computation" hyperplane. In the figure, both arrays use the von Neumann neighborhood index and are two-dimensional. Furthermore, there is a connection between cells in the two arrays. It is intended that each cell in the computation hyperplane be some type of universal logic module that can perform a list of functions. Furthermore, the function to be performed by any given computation cell is determined by its associated cell in the control hyperplane. In this fashion, each computation cell can be "programmed" by the control hyperplane to give the array some particular global (overall) function. Furthermore, if a fault in any computation cell or control cell is detected, it may be possible for the control plane to "reconfigure" such that the faulty cell or cells are not used. This property, if realized, will make the array fault tolerant.

It should be pointed out that the structure of Figure 3.1 represents a particular case of the structure to be considered. The von Neumann neighborhood index is shown only due to its simplicity and the selection of two-dimensional arrays is arbitrary. For this reason, the control hyperplane is best modeled as a tessellation automaton so that arbitrary neighborhoods and array dimensions can be considered. The computation hyperplane can be either sequential or combinational in nature and is permitted to have a neighborhood that differs from that of the control

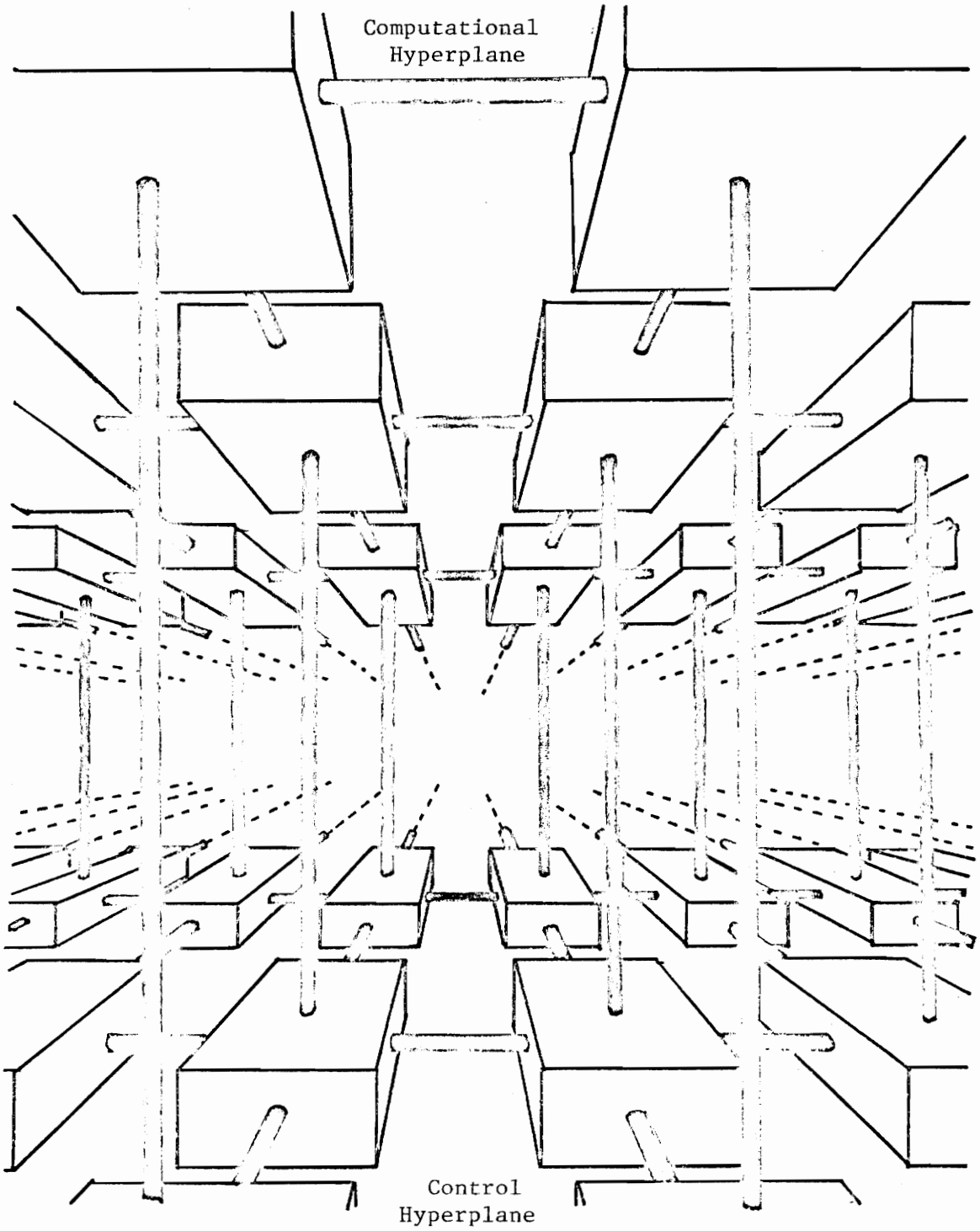


Fig. 3.1 A Cellular Computer

hyperplane. Furthermore, the interconnection of the two hyperplanes is permitted to be a third neighborhood. Then a cell in the computation hyperplane may receive signals from several cells in the control hyperplane to determine the function it is to perform. At this time, no decision will be made as to where the external inputs are provided. Perhaps the control hyperplane receives them, perhaps the computation hyperplane, or perhaps both. If these inputs are supplied to the control hyperplane, then TA = (A, E^d, X, I) is required since the controlling array must make state transitions during computation for the inputs received and the cells of the computational hyperplane can be combinational functions. If the inputs go to the computational array, then TA = (A, E^d, X, σ) is an adequate automaton for the control hyperplane since it will remain in some static global configuration during the computation while the computational cells will be sequential. If inputs are made to a tessellation automaton, they must be provided to every cell in the TA. This must be done so that the cells of the TA can select the correct local transformation for the given input. On first consideration, this seems to be a bad constraint due to the tremendous number of wires that must be run to each cell. However, we claim that this is a necessary condition no matter where the inputs are made. If inputs are made locally to cells, the failure of a cell can make its input unaccessible to the array. The only solution to this difficulty is to provide all inputs to all cells. We do not propose to make any constraints on the structure but intend to consider it with an open mind. The problem of specifying the particular details of the architecture will be solved by finding automata that can do self-reconfiguration.

Before self-reconfiguration can be considered, several other problems must be solved. The first of these is initialization. This refers to the question of how the control hyperplane can be placed in the correct global configuration that corresponds to some desired global function to be computed. Secondly, the problem of self-diagnosis must be examined. By self-diagnosis, we refer to fault detection at the cell level, rather than the circuit level. To do self-diagnosis, the array must first determine that a fault exists and, second, point out the faulty cell or cells. Finally, the question of self-reconfiguration can be approached. This question involves the problem of moving the control hyperplane from one global configuration to another where the cells that have been identified as faulty are not used. In doing so, the new configuration must perform the same computation as the old. There will be a time during reconfiguration that the array cannot accept inputs or make outputs. For this reason, all inputs and outputs should be made in a "handshaking" fashion. If this is done, and outputs are delayed, the array can do exact computations and any faults encountered will only result in a time delay.

The results of this dissertation are concerned primarily with the control hyperplane when viewed as a tessellation automaton. For this reason, the results presented are independent of the cell realization. The choice of a particular realization may have some impact when self-reconfiguration is considered, particularly since the cells of the computation hyperplane are to be universal logic modules. The fact that a cell has failed for one function does not imply that it is entirely

worthless. It is possible that several other functions (perhaps even all other functions) can still be performed correctly with the fault(s) present. Gray and Thompson [46] have considered exactly this question with respect to universal modular trees (UMT). Friedman [43] has shown that any sequential machine can be realized by a sequential machine with a single feedback wire and Arnold [44] applied this result to the UMT. The design of UMT's has been investigated by Nelson [47], Thompson and Gray [48], [49]. The problem of fault detection has been examined by Prasad [39], Prasad and Gray [40], [41], [42], who showed that diagnosis of unrestricted multiple faults within a UMT (and several other iterative structures) can be accomplished. Shih [37], Shih and Gray [38], extended the results of Prasad to include fault location within the UMT for various fault classes. Hoptiak [35], [36] considered a means of implementing results by Prasad on generalized sequential trees. To summarize, the UMT has been shown to be a universal structure and faults within it can be both detected and located. For this reason, it appears to be the most likely candidate for realizing the cells of the two hyperplanes when self-reconfiguration is approached. It should be noted that the UMT is not of mere academic interest. Stewart [20] has shown that rectangular to polar coordinate conversion using CORDIC [50] can be done by an array of relatively simple UMT's. This is not a trivial computation and simple modifications to Stewart's results permit other functions to be computed such as SIN, COS, TAN, LOG, etc. (see [50]).

Gregory [9] has considered the problem of fault detection and isolation at the cell level and considered various techniques for trans-

mitting intercell signals and inputs to and from cells. If the cells are to be synchronous, a very real problem exists concerning the clock. The clock, like everything else, must be "distributed" throughout the array. This problem has been approached by Fletcher, et. al. [51].

Of primary consideration in this dissertation are the problems of initialization and self-diagnosis by the control hyperplane. First, the powerful nature of tessellation automata are considered. Then, results concerning patterns of a combinatorial nature are advanced. These patterns are to be used as global configurations in the tessellation automaton. Next, studies of the rate at which configurations can "grow" are performed. Then, the results found for both patterns and growth rates are combined to solve the initialization problem which we shall refer to as "synthesis". Finally, self-diagnosis is considered for automata with perturbed (faulty) configurations.

IV. TURING MACHINES AND TESSELLATION AUTOMATA

4.1 Introduction

In this chapter, the relationship between Turing machines (TM) and Tessellation Automata (TA) is considered. First, the formal notion of a Turing machine is advanced. Then, a theorem relating them to tessellation automata is introduced which constructs an equivalent TA for any given TM. This proves the existence of a universal TA. It is then shown that the converse is also true: That is, TA can be simulated on Turing machines. Then, the question of stability is addressed and theorems concerning it are introduced.

4.2 Turing Machines

The construction of a Turing machine [23] is shown in Figure 4.2.1 and can be formalized [14] by a five-tuple.

$$Tm = (K, \Gamma, \delta, q_o, F)$$

Where: K is the finite set of states which the controller can assume,

Γ is the finite set of symbols which can exist on the tape,

δ, the "next move function", is a mapping from $K \times \Gamma$ to $K \times \Gamma \times \{L,R\}$,

q_o, an element of K, is the initial state of the controller,

F, a subset of K, are "final" (acceptance) states of the controller.

The subset F is generally utilized when Tm is functioning as a linguistic acceptor (i.e., identifying a string as belonging to a language) and has no application when Tm is used for general computation. Also, Tm should have one or more states which halt (computation completed).

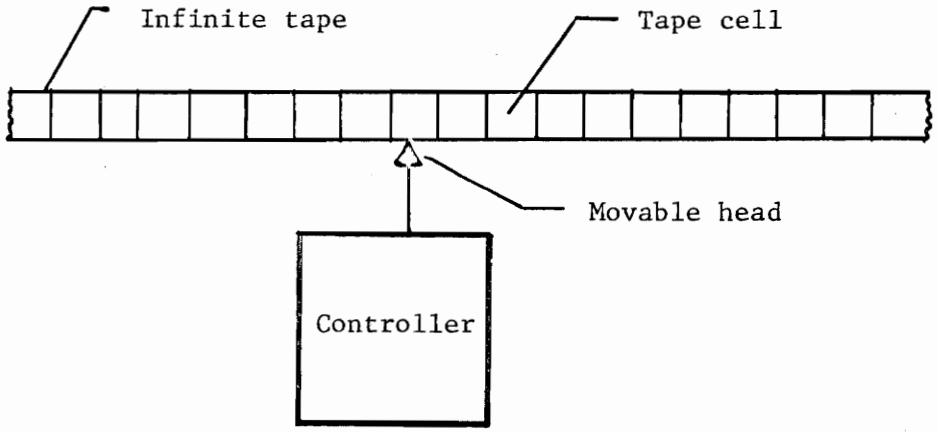


Figure 4.2.1 The Turing Machine Construction

These states can be identified by expanding the head move to include "H" for halt (no move). Further, Tm's definition can be expanded to include the initial tape contents, T_o, at t = 0, and the initial position of the head, p_o, at t = 0. Then T_i denotes the tape content at time i, and, similarly, p_i is the head's position. We shall refer to the content of the tape cell i at time j as $T_j(i)$. These modifications lead to the following six-tuple.

$$Tm = (K, \Gamma, \delta, q_o, T_o, p_o)$$

where δ is modified as stated previously.

$$\delta: \quad (K \times \Gamma) \rightarrow (K \times \Gamma \times \{L,R,H\})$$

This formalization represents a "computational" Turing machine and is no less nor more general than the former definition.

Less formally, the Turing machine simply reads the tape symbol at the current head location and, based on the controller's state and the symbol read, prints a symbol on the tape, goes to a new state, and moves its head left, right, or halts. The tape is assumed to be infinite and T_o, q_o, p_o define the initial configuration of Tm. This definition will be useful later in comparing tessellation automata and Turing machines.

In accordance with Church's hypothesis [8], the Turing machine is the definition of "computable". That is, it is believed to automate <u>any</u> procedure. This makes it the most "powerful" of automata in that any finite sequence of instructions which can be mechanically carried out can be programmed on a Turing machine. Though Church's hypothesis remains unproven, familiarity with Turing machines leads one to accept it. An important fact about Turing machines is that the addition of more tapes and/or more heads does not increase their power. Also,

increases in tape dimensionality do not lead to a more powerful machine. The relevance of these facts is to be considered next.

4.3 Behavioral Equivalence

In this section, the following question is posed. Are Turing machines and tessellation automata computationally equivalent? That is, can a Turing machine be simulated on a tessellation automaton and vice versa? This question has been considered by Burks [6] who states:

"For any given Turing machine M (and hence for a universal Turing machine M_u), there is an initially quiescent automaton in von Neumann's 29-state cellular automaton system which will perform the same computation as M (or M_u)." Thatcher [2] dealt with the particular details for embedding a universal Turing machine in von Neumann's cellular model [25]. It should be noted that von Neumann's model has X = ((0,0), (0,1), (0,-1), (1,0), (-1,0)) and is in E^2. Using this result, Holland's computer [13] can be shown to be universal since it has the same dimensionality and neighborhood index as von Neumann's model and each cell contains a machine at least as powerful as von Neumann's.

We now wish to demonstrate that there exists a class of tessellation automata which are capable of simulating Turing machines but which are one-dimensional in nature. These TA will be equivalent in the following sense. The tape content, head position and controller state will be represented within the cells of the TA as a tuple and the desired information must be "extracted". In terms of computational equivalence, only the tape content need be considered. For that reason, if at any point in time the tape content of Tm can be extracted from

TA, the two automata will be said to be behaviorally equivalent. The following theorem specifies the construction.

Theorem 4.3.1 $\forall$ Tm $\exists$ a behaviorally equivalent TA, TA(Tm) where:

$$TA(Tm) = (A, E^1, (-1,0,1), \sigma)$$

Proof: In TA, let $A = K \times \Gamma \times \{Q,L,R\}$. Then each element $a \in A$ is a three-tuple $a = (k,\ell,s)$ where $k \in K$, $\ell \in \Gamma$, $s \in \{Q,L,R\}$ belong to $Tm = (K, \Gamma, \delta, q_o, T_o, p_o)$ as previously specified. Then, for each move in δ of Tm, $\delta((k,\ell)) = (k',\ell',m)$ where $m \in \{L,R,H\}$ define, in σ, the following:

$$\sigma((\_,\_,Q)(\_,\ell,Q)(k,\_,L)) = (k',\ell',s) \qquad (1)$$

$$\sigma((k,\_,R)(\_,\ell,Q)(\_,\_,Q)) = (k',\ell',s) \qquad (2)$$

Where $s = Q$ if $m = H$, otherwise $s = m$; "_" denotes unspecified or "don't care" condition. These rules cause the TA to make the same moves as Tm. Also in σ must be:

$$\forall k \in K, \ \ell \in \Gamma$$

$$\sigma((\_,\_,L)(k,\ell,Q)(\_,\_,Q)) = (\_,\ell,Q) \qquad (3)$$

$$\sigma((\_,\_,Q)(k,\ell,Q)(\_,\_,R)) = (\_,\ell,Q) \qquad (4)$$

$$\sigma((\_,\_,Q)(k,\ell,L)(\_,\_,Q)) = (\_,\ell,Q) \qquad (5)$$

$$\sigma((\_,\_,Q)(k,\ell,R)(\_,\_,Q)) = (\_,\ell,Q) \qquad (6)$$

$$\sigma((\_,\_,Q)(k,\ell,Q)(\_,\_,Q)) = (\_,\ell,Q) \qquad (7)$$

These rules cause inactive cells to retain the same configuration and force the active cell back to a quiescent state, Q. If T(i) denotes the symbol in the i^{th} cell on the tape, then TA can be initialized as:

$$c(p_o + 1) = (q_o, T_o(p_o + 1), L) \qquad (8)$$

and $\forall_i \neq p_o + 1$,

$$c(i) = (\_, T_o(i), Q) \tag{9}$$

An alternate initialization is:

$$c(p_o - 1) = (q_o, T_o(p_o - 1), R) \tag{10}$$

$\forall_i \neq p_o - 1$,

$$c(i) = (\_, T_o(i), Q) \tag{11}$$

Define a mapping, f: $A \rightarrow \Gamma$ where $\forall k, s$, $f(k,\ell,s) = \ell$. Using f, the tape of Tm can be found from c, the TA's configuration, by applying f to each cell in TA and, $T(i) = f(c(i))$. This mapping merely extracts the tape symbol of Tm from the state-tuple for a given cell in TA. We demonstrate by induction that the preceding construction yields an equivalent TA. The basis for induction is found by going from t = 0 to t = 1 in Tm and its associated TA. At t = 0, we employ (8) and (9) to construct TA(Tm). Then $\forall i$, $i \neq p_o + 1$, $c(i) = (\_, T_o(i), Q)$, and for $i = p_o + 1$, $c(p_o + 1) = (q_o, T_o(p_o + 1), L)$. Since $f(k,\ell,s) = \ell$, then $\forall i$, $f(c(i)) = T_o(i)$. Then at t = 0, the configuration of TA is equivalent to the tape content of Tm. We compute the state of TA and Tm at t = 1. In Tm, $\delta((k,\ell)) = (k',\ell',m)$. Then $\forall i \neq p_o$, $T_1(i) = T_o(i)$, and for $i = p_o$, $T_1(p_o) = \ell'$. In TA, $\forall i \neq p_o$, $c'(i) = c(i)$ by (3), (5) and (7). For $i = p_o$, by (1), $c'(p_o) = (k',\ell',s)$ where s = Q if m = H, otherwise s = m. Since $\forall i \neq p_o$, $f(c'(i)) = \ell$, $f(c'(p_o)) = \ell'$ then $\forall i$, $f(c'(i)) = T_1(i)$. Therefore, the configuration of TA is equivalent to the tape content of Tm at t = 1. A similar argument for the alternate initialization of TA, (10) and (11), can be made. In doing so, rules (2), (4), (6) and (7) are employed. The next state of Tm's controller and its head position are preserved in $c'(p_o) = (k',\ell',s)$. This concludes the

basis of induction. At time t, the tape cells of Tm have content $T_t(i)$, the controller is in some state k, and the head is at a position p_t. We hypothesize that TA(Tm) has $c(i) = (\_,T_t(i),\_)$, $\forall i \neq p_t - 1, p_t, p_t + 1$, and that $c(p_t) = (k,T_t(p_t),Q)$. For cells $p_t - 1$ and $p_t + 1$, either $c(p_t - 1) = (\_,T_t(p_t - 1),R)$, $c(p_t + 1) = (\_,T_t(p_t + 1),Q)$, or $c(p_t - 1) = (\_,T_t(p_t - 1),Q)$, $c(p_t + 1) = (\_,T_t(p_t + 1),L)$. If Tm moved its head right at time t - 1, then the former will be the case. If the head was moved left, it will be the latter case. We shall consider the former case. At time t + 1, $\forall i \neq p_t$, $T_i(t + 1) = T_i(t)$, and $T_{p_t}(t + 1) = \ell'$, the controller goes to a new state k' and the head makes a move, m. This follows from $\delta(k,\ell) = (k',\ell',m)$ given k and p_t. Next, we find the configuration c' of TA. Using (4), (6) and (7), we see that $\forall i \neq p_t$, $c'(i) = (\_,T_t(i),Q)$ and by (2), $c'(p_t) = (k',\ell',s)$ where s = m if $m \neq H$, else s = Q. Then, $\forall i \neq p_t$, $f(c'(i)) = T_t(i)$ and $f(c'(p_t)) = \ell'$. Furthermore, the controller's state at time t + 1 and the move of the head are preserved in $c'(p_t)$. We conclude that the configuration of TA is equivalent to the tape content of Tm at time t + 1. A similar argument using (1), (3), (5) and (7) yields the same result if the latter case of $c(p_t - 1) = (\_,T_t(p_t - 1),Q)$, $c(p_t + 1) = (\_,T_t(p_t + 1),L)$ is considered. Then, by induction, TA(Tm) exists where $TA(Tm) = (A,E^1,(-1,0,1),\sigma)$.

Informally, the basis of the construction is to have only one "active" cell in TA corresponding to the position of the head of Tm. A cell is "active" only if its right neighbor has L in its state-tuple or if its left neighbor has R in its state-tuple. The next state of the active cell records the symbol written by Tm on the tape, the next state

of Tm's controller and activates either its left or right neighbor according to whether Tm moved its head left or right. By considering only the Γ portion of the state-tuple for each cell in TA, the tape of Tm can be reconstructed. An example demonstrating the correspondence is shown in Figure 4.3.1.

We can now prove the universality of E^1, X = (-1,0,1) tessellation automata.

Corollary 4.3.2 There exists a universal tessellation automaton, Ta_u, of the form $TA_\mu = (A,E^1,(-1,0,1),\sigma)$.

Proof: Let Tm_μ be the universal Turing machine. Using Theorem 4.3.1, construct $TA(Tm_\mu)$. Since Tm_μ can simulate any Tm, then by Theorem 4.3.1, $TA(Tm_\mu)$ is also capable of simulating any Tm. Therefore, $TA(Tm_\mu)$ is the universal tessellation automaton, TA_μ, and $TA_\mu = (A,E^1,(-1,0,1),\sigma)$.

The neighborhood index X = (-1,0,1) has been called a Moore neighborhood index by Yamada and Amoroso [30] and is the one-dimensional equivalent of the two-dimensional neighborohood used by Moore [16]. To that end, we shall call TA_μ by the name "Moore's Universal Automaton" in honor of the man who first dubbed these cellular structures "tessellations".

We now wish to consider a previously unaddressed question. Are tessellation automata more powerful than Turing machines? That is, could Church's hypothesis be false? Before addressing this question, some results and definitions due to Yamada and Amoroso [30] are stated.

Theorem 4.3.3 For any TA M_1, there exists a behaviorally isomorphic

Tm

δ: $(A,\not b) \rightarrow (A,\not b,R)$

$(A,X) \rightarrow (B,\not b,L)$

$(B,\not b) \rightarrow (B,X,H)$

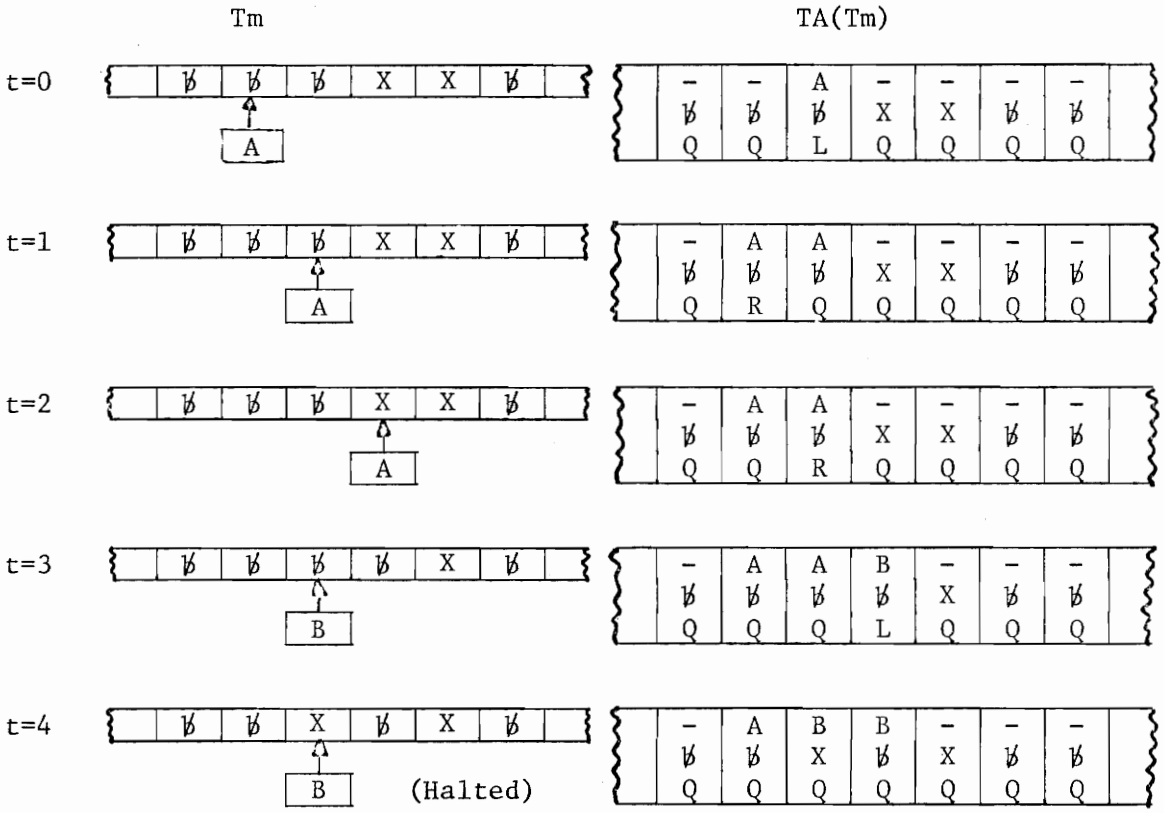


Figure 4.3.1 Construction of TA(Tm)

TA M_2 with a (partial) von Neumann neighborhood index, and M_2 is constructible from a blocked structure (arising from a cover blocking).

A suitable proof can be found in Cole [7]. The d-dimensional von Neumann neighborhood is defined as a (2d + 1)-tuple, $X = (\xi_1, \xi_2, \ldots, \xi_{2d+1})$, where one component is 0^d and all others are of the form $\xi_i = (0, \ldots 0, z, 0, \ldots 0)$ where $z \in \{-1,1\}$. That is, each cell has 2d + 1 neighbors (one being itself) which are directly adjacent in both directions of each dimension. This is a generalization to d-dimensions of von Neumann's two-dimensional neighborhood. (See Figure 2.2 in Chapter II.) A partial von Neumann neighborhood is one which has all its components in von Neumann's neighborhood. It is a subset of a von Neumann neighborhood.

We will also draw on some previously stated results concerning Turing machines. Of particular interest is that any d-dimensional, n-head Turing machine is equivalent to some single tape, single head Turing machine. A formalization of this statement can be found in Arbib [5] and is repeated here.

Theorem 4.3.4 Given an arbitrary Turing machine, Tm, with p tapes, the j^{th} of which is of dimension ℓ_j and is scanned by h_j heads, it may be simulated by a Turing machine, Tm', with a single one-dimensional tape scanned by a single head.

Finally, the questionable power of tessellation automata can be addressed.

Theorem 4.3.5 $\forall$ TA, $\exists$ a behaviorally equivalent Tm(TA) (deterministic) with a single one-dimensional tape scanned by a single head.

Proof: To simulate TA = (A, E^d, X, σ), we require a Turing machine having two d-dimensional tapes, one of which is scanned by n + 1 heads (recall that $|X| = n$) and the other scanned by a single head. In terms of the Tm of Theorem 4.3.4, we require that $p = 2$, $\ell_1 = d$, $\ell_2 = d$, $h_1 = n + 1$ and $h_2 = 1$. The tape having n + 1 heads will be referred to as T_1 and the other (with a single head) as T_2. For such a Turing machine, δ: $(K \times \Gamma^{n+2}) \rightarrow (K \times \Gamma^{n+2} \times M)$ where K is the set of controller states, Γ is the set of tape symbols and M is the set of head moves. Since T_1 and T_2 are d-dimensional, $|M| = 2d$. Since $\Gamma^{n+2} = \Gamma^n \times \Gamma \times \Gamma$, we can restate the next move function of Tm, δ: $(K \times Q \times R \times S) \rightarrow (K \times Q \times R \times S \times M)$ where $Q = \Gamma^n$, $R = \Gamma$, $S = \Gamma$. Let $K = \{0,1\}$ and $\Gamma = A$. Let the heads of Tm be labeled $b_1, b_2, \ldots, b_n, b_{n+1}, b_{n+2}$ where $b_1, \ldots, b_n$, b_{n+1} are on T_1 and b_{n+2} is on T_2. If i is the position of b_{n+2} on T_2, let b_{n+1} be at position i on T_1. Then arrange $b_1, \ldots, b_n$ such that b_j is positioned at $i + \xi_j$ on T_1 where ξ_j is the j^{th} component of X in TA. Let T_1 be initialized such that $T_1 = c_o$ at $t = 0$. Clearly, this is possible since $\Gamma = A$ and T_1, T_2 have the same dimension as TA. Let $q = (q_1, q_2, \ldots, q_n)$ be an element of Q. Then $q \in A^n$ since $Q = \Gamma^n$, $\Gamma = A$. For every move of TA, $\sigma(q) = a$, let $\forall\, x \in A$, $\delta(0,q,x,\_) = (0,q,x,a,m)$. This permits Tm to compute c' of TA onto T_2 without modification of T_1. Also, if $\exists\, j$, $1 \leq j \leq n$, $\ni\, i = i + \xi_j$ then:

$$\forall a \in A,\ \delta(1,(q_1 \ldots q_{j-1} \_ q_{j+1} \ldots q_n), \_, a) =$$
$$(1,(q_1 \ldots q_{j-1}\ a\ q_{j+1} \ldots q_n), a, a, m)$$

Else:

$$\forall a \in A,\ \delta(1, q, \_, a) = (1, q, a, a, m)$$

These rules permit Tm to copy T_2 onto T_1. The choice of rules when K = 1 is due to the possiblity that $\xi_o \in X$. If $\xi_o \in X$, there will be two heads, b_j, b_{n+1} at the same location of T_1. We must insure that they write the same symbol on T_1 if such a case exists. The move, m, is permitted to be non-deterministic. Also, there are two non-deterministic state transitions of the form $\delta(0,q,r,s) = (1, q, r, s, m)$ and $\delta(1, q, r, s) = (0, q, r, s, m)$. Without further formality, we state the principal points. Tm scans T_1 and computes c' of TA onto T_2 where T_1 is initially set equal to c_o of TA. It then moves from K = 0 to K = 1 and copies T_2 onto T_1. Finally, it moves from K = 1 to K = 0 and rescans T_2 where now, T_2 = c'. Clearly, there exists an induction on t showing that T_2 = c for TA for all t. Then, there exists an equivalent Tm(TA) which is non-deterministic. By Theorem 4.3.4, $\exists$ Tm' with one head and a one-dimensional tape. Certainly, Tm' is non-deterministic but it is known [14] that any non-deterministic Tm can be simulated by a deterministic Turing machine. Then $\exists$ Tm(TA) which is deterministic and has a single head scanning a single one-dimensional type.

One final result can be brought forth.

Theorem 4.3.6 $\forall$ TA, $\exists$ a behaviorally equivalent TA = $(A, E^1, (-1,0,1), \sigma)$.

Proof: By Theorem 4.3.5 we can construct Tm(TA) which has one head and a one-dimensional tape. But Tm(TA) can be simulated on TA = $(A, E^1, (-1,0,1), \sigma)$. Therefore, for any TA, we can construct a behaviorally equivalent TA in $(A, E^1, (-1,0,1), \sigma)$.

We can conclude from these two theorems that tessellation automata do not overturn Church's hypothesis and that one-dimensional TA having a Moore neighborhood index are as powerful as any TA. It should be remembered that these simulations cannot (in general) be performed in real time. That is, some two-dimensional TA (in fact, most) can perform an operation in one step that would require vast numbers of steps on their one-dimensional equivalent. For this reason, multidimensional TA are worth considering for certain applications but we argue that they are merely faster, not more powerful.

4.4 Stability of Tessellation Automata

The question to be considered here is similar to the halting question for Turing machines. That is, is it possible to determine if an arbitrary TA in an arbitrary configuration ever reaches equilibrium. The word "equilibrium" is formalized by the following definition.

Definition 4.4.1 A tessellation automaton is said to be in (global) equilibrium if and only if $c' = c$.

Equilibrium is simply the case that the next (global) state of the automaton is equal to the current (global) state. If such a state is reached, clearly the machine will remain in it for all time. The configuration c will be called an equilibrium configuration.

Definition 4.4.2 A tessellation automaton is said to be stable if and only if it eventually reaches equilibrium.

The following well-known theorem is due to Turing [23] and is repro-

duced here for clarity.

Theorem 4.4.1 (The halting question.) There exists no algorithm to determine if an arbitrary Turing machine in an arbitrary configuration will eventually halt.

A suitable proof follows from linguistics [14] and will not be presented here.

In essence, Theorem 4.4.1 says that halting is undecidable for Turing machines as a class. It is, in fact, decidable for particular cases. This point will be of some importance later. The halting theorem has been used to prove other arguments unsolvable [19] just as Turing [23] applied it to the Entscheidungs problem. It will now be invoked once more in this capacity.

Theorem 4.4.2 (Stability.) There exists no algorithm which determines if an arbitrary TA in an arbitrary configuration eventually reaches equilibrium.

Proof: Consider an arbitrary Tm and its corresponding TA(Tm). Clearly, TA(Tm) reaches equilibrium if and only if Tm halts since $c' = c$ if and only if Tm halts. This property is due to σ as follows. If the head of Tm moves left, then some cell in TA(Tm) goes from $(\_,\ell,Q)$ to (k',ℓ',L). [A similar argument for right moves of Tm is $(\_,\ell,Q) \rightarrow (k',\ell',R)$] since this cell has a new state, $c' \neq c$. If, however, Tm halts, then we have $(\_, \ell,Q) \rightarrow (k',\ell',Q)$ which may indeed be a new state (if $\ell' \neq \ell$), but this state is preserved under $\sigma((\_,\_,Q)(k,\ell,Q)(\_,\_,Q)) = (k,\ell,Q)$. Therefore, $c' = c$ and equilibrium of TA(Tm) can be decided if and only if

halting of Tm can be decided. Since there exists no algorithm to determine if Tm halts, there can exist no algorithm to decide if TA(Tm) reaches equilibrium. But ∀Tm,TA(Tm) ε {TA} so equilibrium is undecidable for all TA.

An alternate proof due to Thompson [22] constructs a TA which "solves" Post's Correspondence Problem which has been proven to be unsolvable [19]. This implies that equilibrium for those particular TA is unsolvable.

The following corollary follows directly from Definition 4.4.2 and Theorem 4.4.2.

Corollary 4.4.3 Stability of arbitrary TA is unsolvable.

Although stability is unsolvable in general, it may be solvable for particular automata or even large classes of automata. This same statement is true of Turing machines. For instance, a Turing machine which computes 1 + 2 certainly halts (if correctly programmed!). We now consider such cases.

Lemma 4.4.4 c is an equilibrium configuration if and only if ∀i, $\sigma(c(N(X,i))) = c(i)$.

Proof: Since $c'(i) = \sigma(c(N(X,i)))$, then ∀i, $c' = c(i)$. Therefore, $c' = c$ and c is an equilibrium configuration.

The existence of such a σ is questionable. Consider TA = ({0,1}, E^1, (-1), σ) having $c = \overline{0}1\overline{0}$. For this TA, σ does not exist since for the cell in state "1" we require that $\sigma(0) = 1$ but for all others $\sigma(0) =$

0. Thus, $\overline{0}1\overline{0}$ cannot be an equilibrium configuration for this TA. This does not imply that there are no equilibrium configurations for the TA since $c = \overline{1}$, $[\sigma(1) = 1]$, $c = \overline{0}$, $[\sigma(0) = 0]$ and $c = \ldots010101\ldots$ $[\sigma(0) = 1, \sigma(1) = 0]$ are obvious candidates. The following theorem considers the existence of σ.

Theorem 4.4.5 $\forall c \in C$, $\exists\, \sigma \ni c$ is an equilibrium configuration if $\xi_o \in X$.

Proof: Define σ as: $\forall i$, $\sigma(c(N(X,i))) = c(i)$. Since $\xi_o \in X$, if for some i, j, $c(N(X,i)) = c(N(X,j))$, then $c(i) = c(j)$. Then σ is single valued. Since $\forall i$, $\sigma(c(N(X,i))) = c(i)$, then $c'(i) = c(i)$ and $c' = c$. Therefore, c is an equilibrium configuration.

Notice that the theorem implies only sufficiency. Our next result specifies those $c \in C$ which may be equilibrium configuration if $\xi_o \notin X$.

Theorem 4.4.6 If $\xi_o \notin X$, $\exists\, \sigma \ni c$ is an equilibrium configuration if and only if $\forall i, j \ni c(N(X,i)) = c(N(X,j))$, then $c(i) = c(j)$.

Proof: For equilibrium of c, we require that σ be defined as: $\forall i$, $\sigma(c(N(X,i)) = c(i)$. If $\exists\, i, j, \ni c(N(X,i)) = c(N(X,j))$, $c(i) \neq c(j)$, then σ is multivalued and does not exist. If $\forall i, j \ni c(N(X,i)) = c(N(X,j))$, then $c(i) = c(j)$, σ will be single valued and therefore exists.

Theorem 4.4.5 shows that, if $\xi_o \in X$, we can always define σ for any given configuration such that the configuration is an equilibrium configuration. Theorem 4.4.6 specifies those configurations which can or cannot be made equilibrium configurations if $\xi_o \notin X$. The fact that the

given configuration ever occurs must be demonstrated if the automaton is to be called stable. This question has been proven unsolvable in Theorem 4.4.2 for arbitrary TA, but we must remember that it may be solvable for particular TA or even large classes of TA.

We can demonstrate the existence of a special class of tessellation automata which are completely stable. That is, any configuration placed in the automata is an equilibrium configuration.

Definition 4.4.3 A tessellation automata is called non-erasing if and only if $\forall c \in C$, $c' = c$.

In non-erasing tessellation automata, no modification of the initial configuration will occur since all configurations are equilibrium configurations. This implies that any configuration placed in the automaton remains at rest and is a final configuration. Then the automaton is stable for any configuration.

Theorem 4.4.7 $\forall$TA, $\exists\ \sigma \ni$ TA is non-erasing if and only if $\xi_o \in X$.

Proof: Let $p = a_{\xi_o} a_{\xi_1} \ldots a_{\xi_n}, a_{\xi_i} \in A$, denote an element of A^n. Define σ as: $\forall p \in A^n$, $\sigma(p) = a_{\xi_o}$. Then $\forall c(N(X,i))$, it follows that $\sigma(c(N(X,i)) = c(i)$. Therefore, $c'(i) = c(i)$. Thus, $\forall c \in C$, c is an equilibrium configuration. Then, the TA is non-erasing. If $\xi_o \notin X$, clearly $\exists\ c \ni c(N(X,i)) = c(N(X,j))$ but $c(i) \neq c(j)$. Then TA cannot be non-erasing since, if $\sigma(c(N(X,i)) = c(i)$, then $c'(j) = c(i)$ where $c(i) \neq c(j)$. For equilibrium of c, we require $c'(j) = c(j)$. Then $c' \neq c$ and TA is not non-erasing.

An example of a non-erasing TA is shown in Figure 4.4.1. Automata

$$TA = (\{0,1,2\}, E^1, (-1,0,1), \sigma)$$

σ:
000 - 0
001 - 0
002 - 0
010 - 1
011 - 1
012 - 1
020 - 2
021 - 2
022 - 2
100 - 0
101 - 0
102 - 0
110 - 1
111 - 1
112 - 1
120 - 2
121 - 2
122 - 2
200 - 0
201 - 0
202 - 0
210 - 1
211 - 1
212 - 1
220 - 2
221 - 2
222 - 2

Figure 4.4.1 A Non-Erasing Tessellation Automata

having this property will be employed in synthesis methods of a later chapter. It is important to note that non-erasing TA are stable for any initial configuration.

4.5 Discussion

In order that the findings presented in this chapter be placed in proper perspective with respect to the chapters to follow, the following points should be considered. Since the stability question for tessellation automata is unanswerable in general, it must be investigated on a case by case basis. In fact, there do exist large classes of TA for which stability is always decidable. The non-erasing automata of the previous section are such a class. In a later chapter we will demonstrate the existence of a class of tessellation automata with particular initial configurations for which stability is answerable. Furthermore, the equivalence theorem demonstrates the powerful nature of TA, thus justifying their use in the framework of the application.

V. PATTERNS

5.1 Introduction

In this chapter, we consider the existence of patterns in which all subpatterns of a given size are unique. A one-dimensional pattern (string) having this property is 0001011100. In this string, all substrings of length three are unique. We also note that all possible substrings (000, 001, ..., 111) are in the string. This implies that its length (ten) is maximal if only two symbols, 0,1, are to be employed. The question of primary concern is the converse of the example. That is, how many symbols are required to insure the existence of a pattern (having a given size) whose subpatterns (also of a given size) are unique? The importance of this question will be demonstrated in a later chapter. As a basis for multidimensional patterns, the one-dimensional case of strings is considered first.

5.2 Strings

We now demonstrate the existence of strings similar to that of the previous discussion and find the relation between length of the string and the number of symbols used.

Definition 5.2.1 An (ℓ,n) string is a string of length ℓ having all n-length substrings unique.

It is not required that all possible n-length substrings be contained in the (ℓ,n) string. The previously mentioned string, 0001011100, is a (10,3) string and happens to contain all substrings of length three. The string, 0123456789, is also a (10,3) string but does not contain all

possible substrings of length three.

Definition 5.2.2 An (ℓ,n) string is called cyclic if and only if all rotations of it are (ℓ,n) strings.

In a cyclic (ℓ,n) string, the position of a substring is given modulo-ℓ. For instance, the string 00010111 is a cyclic (8,3) string and the length three substring at location 7 is 110. If an (ℓ,n) string is not cyclic, this is not the case and locations greater than $\ell - n + 1$ do not exist. These strings have been called DeBruijn or Good sequences.

The following Lemma extends a result due to Good [33] to arbitrary symbol set.

Lemma 5.2.1 For a set of symbols, A, $|A| = k$, $\exists$ a cyclic (k^n,n) string.

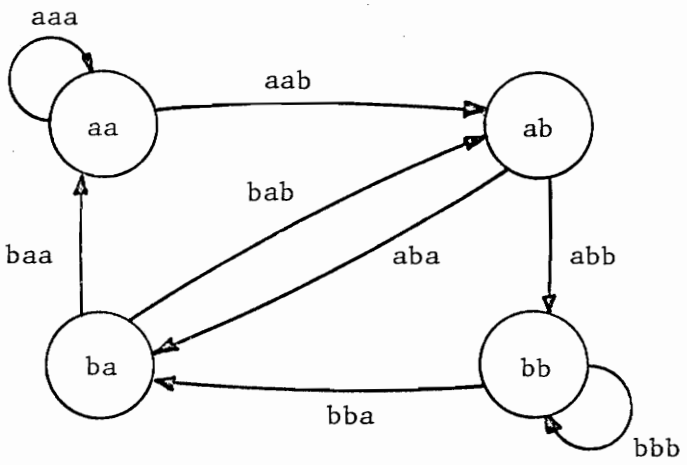
Proof: Construct a directed graph, G, having each vertex, v_i, labeled with a unique substring of length n-1. Since there are k symbols, there will be k^{n-1} vertices. Let edges be labeled with substrings of length n where the first n-1 symbols of the label are called the prefix and the last n-1 symbols are called the suffix. An edge, e_{ij}, is directed from v_i to v_j if and only if the prefix of e_{ij} equals the label of v_i and the suffix of e_{ij} equals the label of v_j. Consider a vertex, v_x, in G. Since the label on v_x has length n-1, there must be exactly k edges directed towards v_x since preceding the label on v_x with any of the k symbols creates a unique label of length n which must be the label of some edge. That is, the label on v_x must be the suffix of k edges. For similar reasons, there are k edges directed away from v_x. This im-

plies that there are k^n edges in G and that the label on each edge is a unique substring of length n. Therefore, the outdegree equals the indegree for all vertices in G. Consider two vertices, v_i, v_j and their labels, $\ell_i = a_1, a_2, a_3, \ldots, a_{n-1}$, $\ell_j = b_1, b_2, \ldots, b_{n-1}$. If b_1 is appended to ℓ_i, the label $a_1, a_2, a_3, \ldots, a_{n-1}, b_1$ is created which must be the label of an edge directed from v_i to another vertex labeled $a_2, a_3, \ldots, a_{n-1}, b_1$. Now b_2 can be appended in a similar fashion to reach a vertex labeled $a_3, \ldots, a_{n-1}, b_1, b_2$. Thus, if the symbols $b_1, b_2, \ldots, b_{n-1}$ are appended to ℓ_1, we reach the vertex, v_j. Therefore, there is a path from v_i to v_j. Since there is a path between any pair of vertices, the graph is connected. Since the graph is connected and the outdegree equals the indegree for every vertex, the graph possesses an Euler circuit (see Liu [15], Corollary 6 - 3.2). Since an Euler circuit passes through every edge once and only once, there exists a string which contains each unique substring of length n once and only once. The string is cyclic since a rotation of the string corresponds to a different starting point in the graph. Since traversing an edge in G corresponds to the addition of one symbol to the string, the length of the string is k^n (there are k^n edges). Then, there exists a cyclic (k^n,n) string.

The construction of G is exemplified in Figure 5.2.1. The cyclic (k^n,n) strings of the Lemma can be made slightly larger if the cyclic property is discarded.

Corollary 5.2.2 For a set of symbols A, $|A| = k$, $\exists$ a $(k^n + n - 1, n)$ string.

A = {a,b}, n = 3, k = 2



Starting at vertex aa, and traversing edges aaa, aab, abb, bbb, bba, bab, aba, baa yields:

abbbabaa

having length $8 = k^n = 2^3$.

Figure 5.2.1 Demonstration of Lemma 5.2.1

Proof: Consider the graph, G, of Lemma 5.2.1. There are k^n edges in G, each of which is labeled with a unique n-length substring. Traverse an Euler circuit in G and construct a string as follows. For the first edge traversed, write the entire n-length label of the edge. For each of the k^n-1 edges traversed thereafter, append only the final symbol of the edge's label. Then, the resulting string has length $n + k^n - 1$ and still has all n-length substrings unique. Thus, it is a $(k^n + n - 1, n)$ string.

The cyclic (8,3) string, abbbabaa, of Figure 5.2.1 can be extended to a (10,3) string, aaabbbabaa, using the Corollary. (In doing so, the initial edge chosen was aaa.) We note that this (10,3) string is not cyclic since its second rotation left is abbbabaaaa which has the subpattern, aaa, in two locations. Although these $(k^n + n - 1, n)$ strings are not cyclic, they are maximal in length.

Definition 5.2.3 An (ℓ,n) string (using k symbols) is called maximal if and only if there exists no (m,n) string (using k symbols), $m > \ell$.

Corollary 5.2.3 $(k^n + n - 1, n)$ strings are maximal.

Proof: We derive the maximal length of an (ℓ,n) string. In a string of length ℓ, there are $\ell - n + 1$ substrings of length n. Given k symbols, there exist k^n unique substrings. If $\ell - n + 1 = k^n$, then $\ell = k^n + n - 1$ and ℓ is maximal since any string of length m, $m > \ell$, must have repeated substrings. Then, the $(k^n + n - 1, n)$ string is maximal.

Shorter (m,n) strings can be obtained from an (ℓ,n) string as is

shown next.

Corollary 5.2.4 Any m-length substring of an (ℓ,n) string is an (m,n) string if and only if $n \leq m \leq \ell$.

Proof: Since an (m,n) string, $m < n$, is an absurdity (its substrings are longer than the string itself) and if $m > \ell$, the (m,n) string cannot be a substring, we consider only $n \leq m \leq \ell$. Let s be a substring of length m, $n \leq m \leq \ell$, of some (ℓ,n) string. Since every substring of length n is unique in the (ℓ,n) string, then every substring of length n in s must also be unique. Therefore, s is an (m,n) string.

Our main result for strings can be stated.

Theorem 5.2.5 $\exists$ an (ℓ,n) string if and only if $k \geq \left\lceil \sqrt[n]{\ell - n + 1} \right\rceil$.

Proof: That $k^n + n - 1$ is the maximal length of an (ℓ,n) string has been shown in Corollary 5.2.3. We merely solve the inequality $\ell \leq k^n + n - 1$ for k.

As stated previously, this result will be of great importance in a later chapter. We can pose a slightly different question to conclude this section. Given ℓ and k, for what n does an (ℓ,n) string exist? Unfortunately, to answer this question in general, we must solve the equation $\ell = k^n + n - 1$ for n (to obtain minimal n) which is transcendental and has no closed form solution. If maximal length of the string is not required, then cyclic (ℓ,n) strings may be considered. For these, $n = \left\lceil \log_k \ell \right\rceil$. This points out the advantage of having the long substrings in that fewer symbols are required to ensure the

existence of an (ℓ,n) string.

Theorem 5.2.6 Any (ℓ,n) string is an (ℓ,m) string if $m \geq n$.

Proof: Since, in the (ℓ,n) string every substring of length n is unique, then every substring of length m is also unique $(m \geq n)$ since each has at least one unique substring of length n contained in it. Therefore, the (ℓ,n) string is also an (ℓ,m) string.

Theorem 5.2.7 Any length p substring of an (ℓ,n) string is a (p,m) string if $n \leq m \leq p \leq \ell$.

Proof: By Theorem 5.2.7, $\exists$ (ℓ,m) string if $m \geq n$. The reduction to a (p,m) string is proven in Corollary 5.2.5.

This concludes our investigation of strings. We proceed by extending these findings to patterns of arbitrary dimension.

5.3 Extended Dimension

The previous section established the basis for similar questions concerning patterns of arbitrary dimension. That is, do there exist multidimensional patterns having all subpatterns (of the same dimension) unique and, if so, how many symbols are required?

Definition 5.3.1 An $\ell(d)$ pattern is a d-dimensional pattern having length ℓ in every dimension.

We will use this same notation to refer to subpatterns. That is, n(d) subpatterns are sub-hypercubes (having length n in all d-dimensions) of some $\ell(d)$ pattern.

Definition 5.3.2 An $(\ell,n)_d$ pattern is a d-dimensional pattern of size ℓ (in each dimension) having all subpatterns of size n (also in each dimension) unique.

Then an $(\ell,n)_d$ pattern is a d-dimensional hypercube that is ℓ x ℓ x ... x ℓ and has all n x n x ... x n subpatterns (also hypercubes) unique.

Definition 5.3.3 An $(\ell,n)_d$ pattern is called cyclic if and only if $\exists$ a dimension, x, $1 \leq x \leq d$ $\ni$ all rotations of the pattern in dimension x are $(\ell,n)_d$ patterns.

Notice that only one such dimension is required. We can extend Lemma 5.2.1 to patterns of arbitrary dimension.

Theorem 5.3.1 For any set of symbols A, $|A| = k$, $\exists$ a cyclic $(k^n,n)_d$ pattern, $n > 1$.

Proof: (by induction on d)

The basis for the induction is proven by Lemma 5.2.1 (for d = 1). We now assume $\exists$ a $(k^n,n)_d$ cyclic pattern and show $\exists$ a $(k^n,n)_{d+1}$ cyclic pattern. Denote as p_1 the $(k^n,n)_d$ pattern which is cyclic in dimension x and let p_r be its r^{th} rotation in dimension x. Since p_1 has length k^n in all dimensions, then $1 \leq r \leq k^n$. Let s(i,j) denote the n(d) subpattern at location i in p_j. Then i is a d-tuple and, since p_j is cyclic, the component of i in dimension x is taken modulo k^n. Since p_1 is a $(k^n,n)_d$ pattern, then $\forall i,j$ $(i \neq j)$, $s(i,1) \neq s(j,1)$. Since p_k is a cyclic rotation of p_1, then $\forall i,j$ $(i \neq j)$, $s(i,k) \neq s(j,k)$. Since p_i and p_j are cyclic rotations of each other, then $\forall i,j$ $(i \neq j)$, $s(k,i) \neq$

$s(k,j)$. Since each cyclic rotation still contains the same subpatterns at some location, then $\forall i,j,k, \exists\ \ell \ni s(\ell,i) = s(k,j)$ and since all k^n rotations are considered $\forall i,j,\ell, \exists\ k \ni s(i,k) = s(j,\ell)$. Let * denote concatenation in dimension d + 1. Then $p_{i_1} * p_{i_2} * \ldots * p_{i_{(k^n)}}$ is a pattern in d + 1 dimensions having size k^n in every dimension. Let x_i be a d-tuple $(\ell_1, \ldots, \ell_{x-1}, i, \ell_{x+1}, \ldots, \ell_d)$ having i at the x^{th} component and consider the sequence, $s(x_1,1), s(x_2,1), \ldots, s(x_{(k^n)},1)$, of subpatterns found in p_1 at locations $(\ell_1, \ldots, \ell_{x-1}, 1, \ell_{x+1}, \ldots, \ell_{(k^n)})$, $(\ell_1, \ldots, \ell_{x-1}, 2, \ell_{x+1}, \ldots, \ell_{(k^n)}), \ldots, (\ell_1, \ldots, \ell_{x-1}, k^n, \ell_{x+1}, \ldots, \ell_{(k^n)})$. (Informally, we are looking at the sequence of subpatterns at a location in p_1 which is arbitrarily fixed in all dimensions except x, the cyclic dimension.) Let $p_{i_1} = p_1$ and select $p_{i_{j+1}}$ where for some r, $s(x_r, i_j) = s(x_1,1)$, $s(x_r,i_{j+1}) = s(x_{j+1},1)$. Since $\forall i,j,k, \exists\ \ell \ni s(\ell,i) = s(k,j)$ and $\forall i,j,\ell, \exists\ k \ni s(i,k) = s(j,\ell)$, then p_{j+1} can always be found. Therefore, the construction exists. Furthermore, $\forall j$ the subpattern $s(x_1,1)$ found in p_{i_j} is paired with $s(x_{j+1},1)$ in $p_{i_{j+1}}$. Then all selections for p_{i_j} are unique. Since $\forall i, j, (i \neq j)$, $s(i,1) \neq s(j,1)$, then $\forall i, (i \neq 1)$, $s(x_i,1) \neq s(x_1,1)$. This implies the pair $s(x_1,1)$, $s(x_{j+1},1)$ is unique only to $p_{i_j} * p_{i_{j+1}}$ for all j. Since $\forall i,j,k\ (i \neq j)$, $s(i,k) \neq s(j,k)$ and $\forall i,j, (i \neq j)$, $s(k,i) \neq s(k,j)$, then $\forall j$, the pair $s(\ell,i_j)$, $s(\ell,i_{j+1})$ is unique only to $p_{i_j} * p_{i_{j+1}}$. Since all pairs of subpatterns in $p_{i_1} * \ldots * p_{i_{(k^n)}}$ are unique in dimension d + 1, then $p_{i_1} * \ldots * p_{i_{(k^n)}}$ is a $(k^n,n)_{d+1}$ pattern if $n > 1$. Clearly, it is cyclic in dimension x since any rotation of it in the x^{th} dimension has the same pairing of subpatterns. Thus, for $n > 1$, $\exists$ a cyclic $(k^n,n)_{d+1}$ pattern. By induction, $\forall d$, $\exists$ a cyclic $(k^n,n)_d$ pattern, $n > 1$.

As a demonstration of Theorem 5.3.1, we extend a string to two and three dimensions. Consider the cyclic string 0011 having k = 2, n = 2, $k^n = 4$. Then, there are four cyclic rotations, p_1 = 0011, p_2 = 0110, p_3 = 1100, p_4 = 1001 and

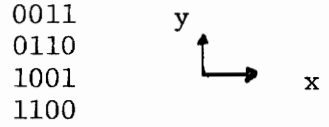
$$s(1,1) = 00$$

$$s(2,1) = 01$$

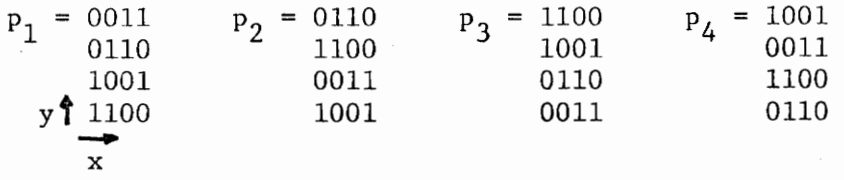
$$s(3,1) = 11$$

$$s(4,1) = 10$$

is the sequence mentioned in the Theorem. Letting $p_{i_1} = p_1$, we see that in p_1, s(1,1) = 00 is at location 1 (numbering from left to right) and we must select for p_{i_2} the string having s(2,1) = 01 in location 1. Then $p_{i_2} = p_2$ and we locate s(1,1) = 00 in p_2 at location 4. Then, for p_{i_3} we must select the pattern having s(3,1) = 11 at location 4. Then $p_{i_3} = p_4$ which has s(1,1) = 00 at location 2, and we select $p_{i_4} = p_3$ since p_3 has s(4,1) = 10 at location 2. Finally, we concatenate p_1 * p_2 * p_4 * p_3 to get:



which is two-dimensional and cyclic with size 4 x 4 having all 2 x 2 size subpatterns unique. It is cyclic in the same dimension as the string. Next, we extend it to three dimensions.



and:

$$\begin{array}{l} \qquad\qquad x,y \\ s((1,1),1) = \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix} \\ s((2,1),1) = \begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix} \\ s((3,1),1) = \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix} \\ s((4,1),1) = \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix} \end{array}$$

Letting $p_{i_1} = p_1$, which has $s((1,1),1)$ at $(1,1)$, we select $p_{i_2} = p_2$ since it has $s((2,1),1)$ at $(1,1)$. Then, in p_2, $s((1,1),1)$ is at $(4,1)$ so we pick $p_{i_3} = p_4$ since p_4 has $s((3,1),1)$ at $(4,1)$. Now, for p_{i_4} we use p_3 because p_4 has $s((1,1),1)$ at $(2,1)$ and p_3 has $s((4,1),1)$ at $(2,1)$. At last, we concatenate $p_1 * p_2 * p_4 * p_3$ to find a cyclic 4 x 4 x 4 pattern with unique 2 x 2 x 2 subpatterns.

```
                1 1 0 0
                1 0 0 1
                0 1 1 0
                0 0 1 1

            1 0 0 1
            0 0 1 1
            1 1 0 0
            0 1 1 0

        0 1 1 0
        1 1 0 0
        0 0 1 1
        1 0 0 1

    0 0 1 1
    0 1 1 0
    1 0 0 1
    1 1 0 0
```

It can be shown that the patterns of Theorem 5.3.1 are cyclic in every dimension if a certain adjacency between p_{i_1} and $p_{i_{(k^n)}}$ is defined.

Corollary 5.3.2 For any set of symbols, A, $|A| = k$, $\exists$ a $(k^n,n)_d$ pattern, $n > 1$, which is cyclic in every dimension.

Proof: The induction of Theorem 5.3.1 will prove the corollary if it is shown that $p_{i_1} * \ldots * p_{i_{(k^n)}}$ is cyclic in dimension d + 1. We note that $\forall$ j, s(j,1) is not paired with s(j,1) in the concatenation. Since $\exists$ ℓ $\ni$ $s(\ell,i_{(k^n)}) = s(x_1,1)$ and since $p_{i_{(k^n)}}$ is a rotation of p_1 in dimension x, then ℓ and x_1 differ only in the x^{th} component. Let $h = \ell - x_1$ and define the adjacency between p_{i_1} and $p_{i_{(k^n)}}$ where $\forall$j, location j in $p_{i_{(k^n)}}$ is adjacent to location j - h in p_{i_1}, j - h taken modulo k^n. This is a simple rotation in dimension x when going from $p_{i_{(k^n)}}$ to p_{i_1}. It is clear that $s(x_1,1)$ found in $p_{i_{(k^n)}}$ (at location ℓ) is now adjacent to $s(x_1,1)$ found in p_{i_1} (at location x_1) since $\ell - h = \ell - (\ell - x_1) = x_1$. Since h merely rotates the adjacency between $p_{i_{(k^n)}}$ and p_{i_1}, it is clear that $\forall$j, s(j,1) found in $p_{i_{(k^n)}}$ is paired with s(j,1) found in p_{i_1}. Since these pairs do not appear in $p_{i_1} * \ldots * p_{i_{(k^n)}}$, then it follows that $p_{i_1} * \ldots * p_{i_{(k^n)}}$ is cyclic in dimension d + 1. Then, by induction, $\exists$ a $(k^n,n)_d$ pattern, $n > 1$, which is cyclic in all dimensions.

To demonstrate the corollary, we consider the $(4,2)_2$ pattern of the previous example.

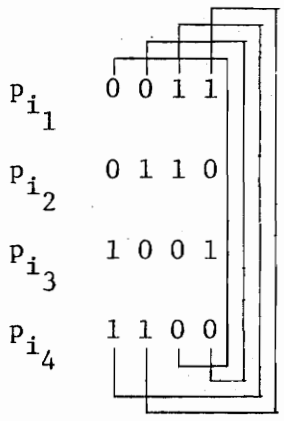
p_{i_1} 0 0 1 1

p_{i_2} 0 1 1 0

p_{i_3} 1 0 0 1

p_{i_4} 1 1 0 0

In this case, $s(x_1,1) = 00$ is found at location 3 in p_{i_4}. Then h = 3 - 1 = 2. The adjacency between p_{i_4} and p_{i_1} is shown below.



This is a rotational matching between p_{i_4} and p_{i_1}. If the pattern is rotated "up" we find:

0 1 1 0
1 0 0 1
1 1 0 0
1 1 0 0

which is still a $(4,2)_2$ pattern. The second rotation "up" is:

1 0 0 1
1 1 0 0
1 1 0 0
1 0 0 1

which is also $(4,2)_2$. It should be clear that under rotation in the new dimension, the pairing of unique subpatterns remains the same. Then any rotation will be a $(4,2)_2$ pattern.

We shall present an application for such patterns in a later chapter. We note that if h = 0, we have the usual cyclic adjacency in the new dimension, d + 1. That is, no rotation in dimension x is required when going from p_{i_1} to $p_{i_{(k^n)}}$ or vice versa. This implies that $p_{i_1} = p_{i_{(k^n)}}$. Some effort has been expended to find such patterns. None has been found for k = n = 2 but for k = 3, n = 2 the following pattern has the property.

A A B A C B B C C

A B A C B B C C A

A C B B C C A A B

B C C A A B A C B

A B A C B B C C A

B C C A A B A C B

A C B B C C A A B

A B A C B B C C A

A A B A C B B C C

It is a $(9,2)_2$ pattern which is cyclic in all dimensions and has h = 0. Due to their cumbersome nature, larger patterns have not been investigated. Since the application to be presented does not require h = 0, we leave this as an open question. Furthermore, since the method employed uses a specific construction, there may be a different method which finds $(k^n,n)_d$ patterns which are cyclic in every dimension with h = 0. If some particular application requires h = 0, additional symbols could be used to supplement the alphabet. In this way an $(\ell,n)_d$ pattern might be found for h = 0. Clearly, this approach would use excessive symbols to achieve its result.

The next result considers the case where n = 1.

Theorem 5.3.3 For any set of symbols, A, $|A| = k$, $\exists$ a cyclic $(\ell,1)_d$ pattern if and only if $k \geq \ell^d$.

Proof: If all n(d) size subpatterns are unique, n = 1, then all 1 x 1 x ... x 1 subpatterns are unique. This implies that every symbol is unique and, since there are ℓ^d locations in an $\ell(d)$ pattern, then $k = \ell^d$ is the minimal k. Therefore, $k \geq \ell^d$ is sufficient. Such a pattern is clearly cyclic.

As in the case for strings, if the cyclic property is discarded, the pattern can be made slightly larger.

Corollary 5.3.4 For any set of symbols, A, $|A| = k$, $\exists$ a $(k^n + n - 1, n)_d$ pattern.

Proof: (by induction on d)
The basis of the induction is established in Lemma 5.2.1 and Corollary 5.2.2 for d = 1. We have shown the existence of cyclic $(k^n, n)_d$ patterns in Theorem 5.3.1. We presume $\exists$ a $(k^n + n - 1, n)_d$ pattern and demonstrate the existence of a $(k^n + n - 1, n)_{d+1}$ pattern. Consider the concatenation of Theorem 5.3.1. If each $(k^n, n)_d$ cyclic pattern is extended to a $(k^n + n - 1, n)_d$ pattern (not cyclic) before concatenation, then the resulting pattern has length $k^n + n - 1$ in all dimensions except d + 1. Consider the subpattern $x(x_1, 1)$. It is paired with $s(x_2, 1)$, $s(x_3, 1)$, ..., $s(x_{(k^n)}, 1)$ in the concatenation but the pair $s(x_1, 1)$, $s(x_1, 1)$ has not been used. If p_{i_1} is concatenated n - 1 times and precedes $p_{i_1} * \ldots * p_{i_{(k^n)}}$, then the resulting pattern has length $k^n + n - 1$ in all dimensions. Since all n(d + 1) subpatterns are still unique, then the pattern is a $(k^n + n - 1, n)_{d+1}$ pattern. Then, by induction,

$\exists$ a $(k^n + n - 1, n)_d$ pattern.

Theorem 5.3.5 $\exists$ an $(\ell,n)_d$ pattern if $k \geq \lceil \sqrt[n]{\ell - n + 1} \rceil$.

The proof of this theorem follows from the same argument in Theorem 5.2.5.

As may not be expected, these patterns are not maximal. By considering the argument of Corollary 5.2.4 extended to $(\ell,n)_d$ patterns, we realize that for maximal patterns, $k^{(n^d)} = (\ell - n + 1)^d$. The length can be expressed as

$$\ell = k^{(\frac{n^d}{d})} + n - 1$$

If we consider the case that $d = 2$, $n = 3$, $k = 2$, we find $\ell = 24.63$ and can conclude that such a pattern does not exist. This does not preclude the existence of a non-hypercube pattern such as a 32 x 16 or 64 x 8 pattern. In fact, the bound does exist if d divides n^d. This fact alone does not prove the existence of such a pattern. We have shown the existence of patterns having $\ell = k^n + n - 1$ which are maximal if $n^d = nd$.

5.4 Irregular Patterns

The previous sections restrict their attention to hypercubic patterns. We now address the problem of patterns which are $\ell_1 \times \ell_2 \times \ldots \times \ell_d$ size having all $n_1 \times n_2 \times \ldots \times n_d$ size subpatterns unique.

Def. 5.4.1 An $\overline{\ell}(d)$ pattern is a hypercube of length ℓ_i in dimension i.

Then $\overline{\ell}$ is a d-tuple, $\overline{\ell} = (\ell_1, \ell_2, \ldots, \ell_d)$ where each component in $\overline{\ell}$ specifies the length of the pattern in that dimension.

Definition 5.4.2 An $(\bar{\ell},\bar{n})_d$ pattern is an $\bar{\ell}(d)$ pattern with all $\bar{n}(d)$ subpatterns unique.

In considering $(\bar{\ell},\bar{n})_d$ patterns, we see that solving for k in:

$$k^{\left(\frac{\prod_{i=1}^{d} n_i}{d}\right)} \geq \prod_{i=1}^{d} (\ell_i - n_i + 1)$$

yields the sufficient number of symbols but we cannot presume the pattern's existence. Although the resultant will not utilize minimal k, $(\ell,n)_d$ patterns may be employed. First, we formulate some preliminaries.

Lemma 5.4.1 Any m(d) subpattern of an $(\ell,n)_d$ pattern is an (m,n) subpattern if $n \leq m \leq \ell$.

Proof: Consider an $(\ell,n)_d$ pattern and an arbitrary m(d) subpattern, $n \leq m \leq \ell$, in it. If some n(d) subpattern in the m(d) subpattern is not unique, then it is also not unique in the $(\ell,n)_d$ pattern. But all n(d) subpatterns in any $(\ell,n)_d$ pattern are unique by definition. Then all n(d) subpatterns of the m(d) subpattern are unique. If $m > \ell$, $\nexists$ m(d) subpattern in an $(\ell,n)_d$ pattern and an $(m,n)_d$ pattern is an absurdity if $m < n$. We conclude that the m(d) subpattern is an $(m,n)_d$ pattern if $n \leq m \leq \ell$.

The Lemma considers reduction in size of $(\ell,n)_d$ patterns. Next, we consider increases in subpattern size.

Lemma 5.4.2 Any $(\ell,n)_d$ pattern is an $(\ell,m)_d$ pattern if $m \geq n$.

Proof: Since every m(d) subpattern of an $(\ell,m)_d$ pattern has at least one n(d) subpattern within it (if $m \geq n$), we conclude that all m(d) subpatterns are unique because all n(d) subpatterns are unique. Then the $(\ell,n)_d$ pattern is an $(\ell,m)_d$ pattern.

Finally, we summarize the two in a single theorem.

Theorem 5.4.3 Any $(\ell,n)_d$ pattern is a $(p,m)_d$ pattern if $n \leq m \leq p \leq \ell$.

Proof: Reduction in size from $(\ell,n)_d$ to $(p,n)_d$ has been shown in Lemma 5.4.1 and requires $n \leq p \leq \ell$. If $m > p$, a $(p,m)_d$ pattern is absurd so we stipulate $m \leq p \leq \ell$. Lemma 5.4.2 permits increased subpattern size from $(p,n)_d$ to $(p,m)_d$ if $m \geq n$.

The following definitions will prove convenient for later results.

Definition 5.4.3 The largest component, ℓ_i, in $\overline{\ell}$, $\overline{\ell} = (\ell_1, \ell_2, \ldots, \ell_d)$ is denoted as MAX $\overline{\ell}$. That is, $\forall j$, $1 \leq j \leq d$, then $\ell_i \geq \ell_j$.

Definition 5.4.4 The smallest component, ℓ_i, in $\overline{\ell}$, $\overline{\ell} = (\ell_1, \ell_2, \ldots, \ell_d)$ is denoted as MIN $\overline{\ell}$. That is, $\forall j$, $1 \leq j \leq d$, then $\ell_i \leq \ell_j$.

We shall employ these conventions in discussions of $\overline{n}$ as well.

Corollary 5.4.4 Any $(\ell,n)_d$ pattern is an $(\overline{\ell},n)_d$ pattern if MAX $\overline{\ell} \leq \ell$, MIN $\overline{\ell} \geq n$.

Proof: By the same argument as Lemma 5.4.1, the corollary is proven.

Corollary 5.4.5 Any $(\ell,n)_d$ pattern is an $(\ell,\overline{n})_d$ pattern if MIN $\overline{n} \geq n$.

Proof: By the same argument as Lemma 5.4.2, the corollary is proven.

Corollary 5.4.6 Any $(\ell,n)_d$ pattern is an $(\overline{\ell},\overline{n})_d$ pattern if MAX $\overline{\ell} \leq \ell$, MIN $\overline{n} \geq n$.

We presume the $(\overline{\ell},\overline{n})_d$ pattern is not absurd. That is, $\forall i$, $\ell_i \geq n_i$.

Now, we can compute the number of symbols required to construct an $(\overline{\ell},\overline{n})_d$ pattern. As stated previously, the value computed will be sufficient but may not be necessary.

Theorem 5.4.7 $\exists$ an $(\overline{\ell},\overline{n})_d$ pattern if

$$k \geq \left\lceil \sqrt[\text{MIN } \overline{n}]{\text{MAX } \overline{\ell} - \text{MIN } \overline{n} + 1} \right\rceil$$

Proof: By Corollary 5.3.4, $\exists$ a $(k^n + n - 1, n)_d$ pattern which, by Theorem 5.3.5 requires

$$k \geq \left\lceil \sqrt[n]{\ell - n + 1} \right\rceil$$

If ℓ = MAX $\overline{\ell}$, n = MIN $\overline{n}$, then by Corollary 5.4.6 $\exists$ an $(\overline{\ell},\overline{n})_d$ pattern. Then,

$$k \geq \left\lceil \sqrt[\text{MIN } \overline{n}]{\text{MAX } \overline{\ell} - \text{MIN } \overline{n} + 1} \right\rceil.$$

This Theorem is our main result for non-hypercubic patterns and concludes our formal investigation of patterns.

5.5 An Example

In order that the theorems of the previous sections be confirmed, we shall construct a two-dimensional pattern that is 7 x 9 having all

4 x 3 subpatterns unique. Formally, this is a $((7,9),(4,3))_2$ pattern. For this pattern, MAX $\overline{\ell}$ = 9, MIN $\overline{n}$ = 3. Applying Theorem 5.4.7, we see that:

$$k \geq \lceil \sqrt[3]{9 - 3 + 1} \rceil = \lceil 1.91 \rceil = 2$$

is the sufficient number of symbols. We begin with Lemma 5.2.1, using two symbols, {0,1}, and construct the graph of Figure 5.5.1. In doing so, we find a cyclic string of length $k^{\text{MIN } \overline{n}} = 2^3 = 8$. The string found is 01110100, which may be extended using Corollary 5.2.2 to length $k^{\text{MIN } \overline{n}}$ + MIN $\overline{n}$ - 1 = 8 + 3 - 1 = 10. The extended string is 0111010001. Next, we take all rotations of 01110100 and extend them in a similar manner.

0 1 1 1 0 1 0 0 → 0 1 1 1 0 1 0 0 0 1 = p_1

1 1 1 0 1 0 0 0 → 1 1 1 0 1 0 0 0 1 1 = p_2

1 1 0 1 0 0 0 1 → 1 1 0 1 0 0 0 1 1 1 = p_3

1 0 1 0 0 0 1 1 → 1 0 1 0 0 0 1 1 1 0 = p_4

0 1 0 0 0 1 1 1 → 0 1 0 0 0 1 1 1 0 1 = p_5

1 0 0 0 1 1 1 0 → 1 0 0 0 1 1 1 0 1 0 = p_6

0 0 0 1 1 1 0 1 → 0 0 0 1 1 1 0 1 0 0 = p_7

0 0 1 1 1 0 1 0 → 0 0 1 1 1 0 1 0 0 0 = p_8

Then, by Corollary 5.3.4 we concatenate $p_1 * p_1 * p_1 * p_2 * p_4 * p_7 * p_3 * p_8 * p_6 * p_5$ to obtain a $(10,3)_2$ pattern.

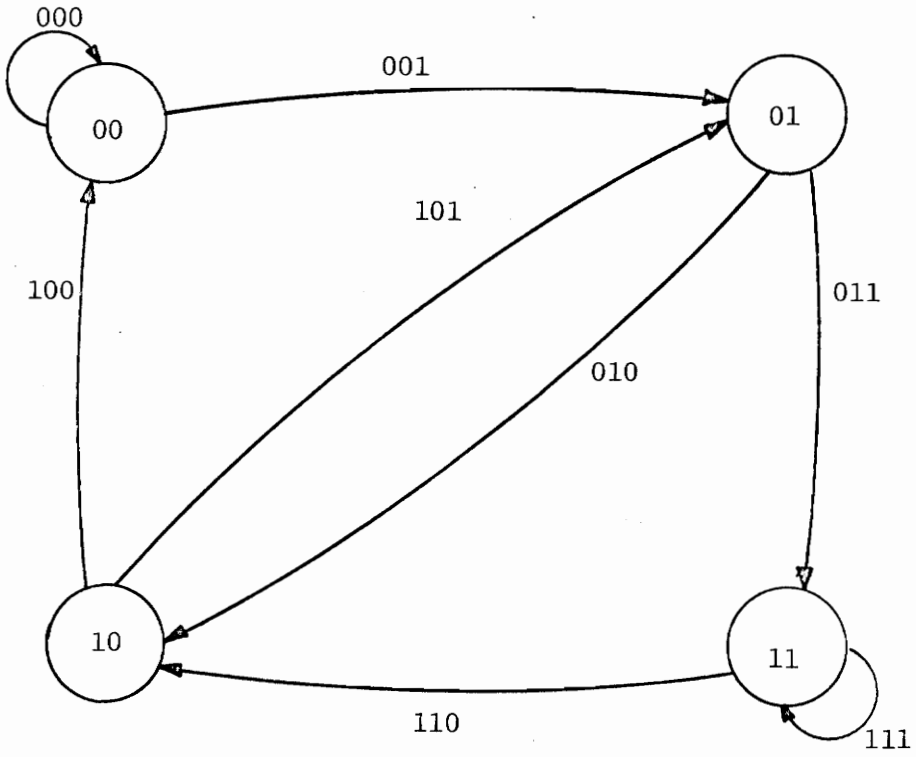


Figure 5.5.1 Finding a Cyclic String

0 1 1 1 0 1 0 0 0 1

0 1 1 1 0 1 0 0 0 1

0 1 1 1 0 1 0 0 0 1

1 1 1 0 1 0 0 0 1 1

1 0 1 0 0 0 1 1 1 0

0 0 0 1 1 1 0 1 0 0

1 1 0 1 0 0 0 1 1 1

0 0 1 1 1 0 1 0 0 0

1 0 0 0 1 1 1 0 1 0

0 1 0 0 0 1 1 1 0 1

Then, the desired pattern can be found by taking any 7 x 9 subpattern of this $(10,3)_2$ pattern. Such a subpattern is:

0 1 1 1 0 1 0 0 0

0 1 1 1 0 1 0 0 0

0 1 1 1 0 1 0 0 0

1 1 1 0 1 0 0 0 1

1 0 1 0 0 0 1 1 1

0 0 0 1 1 1 0 1 0

1 1 0 1 0 0 0 1 1

Since, in this 7 x 9 pattern, all 3 x 3 subpatterns are unique, we can conclude that all 4 x 3 subpatterns are also unique. Thus, we have found a $((7,9),(4,3))_2$ pattern. We can consider the minimal k for such a pattern. We stated that:

$$k^{(\prod_{i=1}^{d} n_i)} \geq \prod_{i=1}^{d} (\ell_i - n_i + 1)$$

and for our case,

$$k^{(3)(4)} \geq (7 - 4 + 1)(9 - 3 + 1)$$

$$k^{12} \geq (4)(7)$$

$$k \geq \lceil \sqrt[12]{28} \rceil = \lceil 1.32 \rceil$$

$$k = 2$$

Then, we have met the minimal k since we employed only two symbols. Of course, this may not always be the case.

This concludes our investigation of patterns. The main results lie in Corollary 5.3.4 which demonstrates the existence of $(k^n + n - 1, n)_d$ patterns and Theorem 5.3.5 which finds the number of symbols required to construct them. Although these results may not lead to minimal k, we will make use of them in a later chapter.

VI. GROWTH

6.1 Introduction

We wish to consider the minimum number of time steps required for the tessellation automaton to move from an initial configuration which is, perhaps, primitive, to a desired configuration. By considering only the minimum time required, we are able to momentarily cast aside the question of stability. Whether or not the automaton halts at the desired configuration is unimportant for now. We are merely stating that it cannot possibly reach the desired configuration in a time less than the minimum found in this chapter. In a following chapter, we shall demonstrate a technique which insures stability of the automaton with respect to the desired configuration. First, some definitions and preliminary results which are of an arbitrary nature with respect to dimension and neighborhood are presented.

Definition 6.1.1 A cell, i, in a configuration, c, having $c(N(X,i)) \neq 0^n$ is called an initializable cell.

Definition 6.1.2 A local transformation, σ, is called maximal with respect to a configuration, c, if and only if $\forall i \ni c(N(X,i)) \neq 0^n$, then $\sigma(c(N(X,i))) \neq 0$.

Informally, a local transformation which is maximal with respect to a configuration, c, will cause all initializable cells to have a non-quiescent state (i.e. $\neq 0$) in c', the next configuration. In other words, it initializes all initializable cells in c.

Definition 6.1.3 A local transformation is called maximal with respect

to a sequence, $c_{i_1}, c_{i_2}, \ldots, c_{i_k}, \ldots, c_{i_j}$, ($\forall k$, $1 \leq k \leq j-1$, $c_{i_{k+1}} = \tau(c_{i_k})$) if and only if $\forall k$, $1 \leq k \leq j$, σ is maximal with respect to c_{i_k}.

By our definition, a local transformation which is maximal with respect to a sequence is maximal for each configuration at each step in the sequence.

Theorem 6.1.1 A local transformation, σ, is maximal with respect to any sequence if and only if $\forall p \in A^n$, $p \neq 0^n$, $\sigma(p) \neq 0$.

Proof: Since every symbol, s, in any arbitrary configuration, c, is in A, then $\forall i$, c, $c(N(X,i)) \in A^n$. But $\forall p \in A^n$, $p \neq 0^n$ we require that $\sigma(p) \neq 0$. Then $\forall$ $c(N(X,i))$, it follows that $\sigma(c(N(X,i))) \neq 0$. Thus, σ is maximal for all configurations, $c \in C$. Then, it is maximal for any sequence since C is the set of all configurations. This demonstrates sufficiency. Clearly, $\exists\, c \in C \ni \forall p \in A^n$, $p = c(N(X,i))$ for some i. (Such a configuration can be found for E^1 TA by embedding a $(k^n + n - 1, n)$ string in a quiescent infinite tessellation. Recall that $(k^n + n - 1, n)$ strings contain all substrings of length n.) Let c be the initial configuration in some TA. If $\exists\, p \in A^n$, $\sigma(p) = 0$, then $\exists i \ni \sigma(c(N(X,i))) = 0$. By Definition 6.1.2, σ is not maximal for the given c. Then $\exists$ a sequence for which σ is not maximal. Thus, necessity is proven.

The Theorem provides an easy check for maximality of the local transformation but the conditions stated are necessary only if any arbitrary sequence is permitted.

Corollary 6.1.2 A local transformation, σ, is maximal with respect to a

sequence if $\forall p \in A^n$, $p \neq 0^n$, $\sigma(p) \neq 0$.

The Corollary follows from the sufficiency of the Theorem. For a given sequence, the condition of the Corollary is not necessary since there may exist some element, q, $q \in A^n$, which never appears as some c(N(X,i)) in any configuration in the sequence. For such an element, if $\sigma(p) = 0$, then σ may still be maximal in every sense. Since synthesis will consist of a predetermined sequence, the Theorem and its Corollary will not be employed and it is included only for completeness.

Definition 6.1.4 Two cells i,j are called inverse neighbors if and only if $(i-j) \in X$.

Lemma 6.1.3 For a cell, i, $\exists$ n inverse neighbors.

Proof: Since $|X| = n$, $\exists\ \xi_1, \xi_2, \ldots, \xi_n$. Then cells $j_1, j_2, \ldots, j_n$ are inverse neighbors of cell i where $j_k = i - \xi_k$, $1 \leq k \leq n$.

These are the neighbors which receive the state information of cell i. They are found by a "backwards" trace.

Lemma 6.1.4 In a primitive configuration, c_p, $\exists$ n initializable cells.

Proof: Recall that a primitive configuration is of the form, $c_p = \overline{0}a\overline{0}$, $a \in A$. That is, $\exists$ a unique cell, i, where $c(i) \neq 0$. For the n inverse neighbors of cell i, (cells $j_1, j_2, \ldots, j_n$), then $c(N(X,j_k)) \neq 0^n$ $(1 \leq k \leq n)$ and for all other cells, j, $(\nexists k \ni j = j_k)$; then $c(N(X,j)) = 0^n$. Therefore, there are exactly n initializable cells.

6.2 One-dimensional Growth

In this section, growth from an initial configuration is considered. First, definitions suitable only for one-dimensional automata are presented. These will be extended shortly to include multidimensional cases.

Definition 6.2.1 If and only if $\exists\, n > 0, k, \ni X = ((k + 1),(k+2), \ldots, (k + n))$, then X is called scope-n.

The definition is due to Yamada-Amoroso [29]. Notice that any scope-n neighborhood is contiguous.

Definition 6.2.2 If and only if $\exists\, i, \ell, \ni \forall j, i + 1 \leq j \leq i + \ell, c(j) \neq 0$, then c is called scope-ℓ and is said to be at location i.

The definition simply extends the concept of scope to configurations. Using these results and definitions, we find our results for one-dimensional tessellation automata.

Lemma 6.2.1 If X is scope-n, c is scope-r, σ is maximal with respect to c, then c' is scope-[r + n - 1].

Proof: Since c is scope-r, then $c = \overline{0}\, a_{i_1}\, a_{i_2} \ldots a_{i_j} \ldots a_{i_r}\, \overline{0}$ where $\forall j, 1 \leq j \leq r, a_{i_j} \in A, a_{i_j} \neq 0$. Denote the position of the initialized cells in c as $p_1, p_2, \ldots, p_j, \ldots, p_r$ where $c(p_1) = a_{i_1}, c(p_2) = a_{i_2}, \ldots, c(p_r) = a_{i_r}$. Note that $p_2 = p_1 + 1, p_3 = p_1 + 2, \ldots, p_r = p_1 + r$. Then, in general, $p_j = p_1 + j - 1, 1 \leq j \leq r$. Also, recall that if X is scope-n, then $X = ((k + 1), (k + 2), \ldots, (k + n))$. Consider the initializable cells associated with cell p_j. They are,

$p_j - (k+1)$, $p_j - (k+2)$, ..., $p_j - (k+n)$, but these positions may be expressed as $(p_1 + j - 1 - k - 1)$, $(p_1 + j - 1 - k - 2)$, ..., $(p_1 + j - 1 - k - n)$, which is $(p_1 + j - k - 2)$, $(p_1 + j - k - 3)$, ..., $(p_1 + j - k - (n+1))$. Writing these in ascending order yields $(p_1 + j - k - (n+1))$, $(p_1 + j - k - (n+1) + 1)$, ..., $(p_1 + j - k - (n+1) + (n-1))$. Let $p_1 - k - (n+1) = p$. Then the initializable cells associated with p_j are $(p + j)$, $(p + j + 1)$, ..., $(p + j + (n-1))$. In a like manner, the initializable cells associated with p_{j+1} are $(p + j + 1)$, $(p + j + 2)$, ..., $(p + j + n)$. Of these cells, only $(p + j + n)$ is not already specified as initializable by p_j. Then, for $1 \leq j \leq r$, we conclude that the initializable cells are at positions $(p + 1)$, $(p + 2)$, ..., $(p + r + (n - 1))$. Since σ is maximal with respect to c, then each of these cells is initialized. Therefore, c' is scope-$[r + n - 1]$.

We now find the scope of the configuration after an arbitrary number of steps.

Lemma 6.2.2 If X is scope-n, c_0 is scope-r, σ is maximal with respect to the sequence $c_0, c_1, c_2, \ldots, c_{s-1}$, then c_s is scope-$[r + s(n-1)]$.

Proof: Consider c_1. By Lemma 6.2.1, c_1 is scope-$[r + (n - 1)]$. We hypothesize that c_s is scope-$[r + s(n - 1)]$ and show c_{s+1} is scope-$[r + (s + 1)(n - 1)]$. Let c_s be scope-p, $p = r + s(n - 1)$. Then, by Lemma 6.2.1, c_{s+1} is scope-k, $k = p + (n - 1) = r + s(n - 1) + (n - 1) = r + (s + 1)(n - 1)$. Then, by induction, the Lemma is proven.

Our main result for one-dimensional automata follows from this lemma.

Theorem 6.2.3 If X is scope-n, c_0 is scope-r, σ is maximal with respect to the sequence $c_0, c_1, \ldots, c_{t-1}$, then the minimum number of steps, t, to reach c, c is scope-ℓ, is $t = \left\lceil \frac{\ell - r}{n - 1} \right\rceil$.[1]

Proof: To reach c in minimal time, we require c_s to be scope-p, $p \geq \ell$ and c_{s-1} to be scope-q, $q < \ell$. By Lemma 6.2.2, $p = r + s(n - 1)$. Thus, we solve the inequality $\ell \leq r + s(n - 1)$ for s which yields $s \geq \frac{\ell - r}{n - 1}$. Since t must be an integer, the minimal value is $t = \left\lceil \frac{\ell - r}{n - 1} \right\rceil$.

The result may be particularized to growth from a primitive configuration.

Corollary 6.2.4 If X is scope-n, $c_0 = c_p$, σ is maximal with respect to the sequence $c_0, c_1, \ldots, c_{t-1}$, then the minimum number of steps, t, to reach c, c is scope-ℓ, is $t = \left\lceil \frac{\ell - 1}{n - 1} \right\rceil$.

We note that, if n = 1, our result yields $t = \infty$ as the minimal time to reach c. This is a correct result since, if n = 1, there is only one element in X. Thus, for each initialized cell in c, there is a single initializable cell. Then c and c' have equal scope and growth is not possible. The configuration is not necessarily at equilibrium since it may shift in one direction or another. We also note that if $\ell = 1$ then we see that t = 0. This, too, is correct since the primitive configuration which is the automaton's initial configuration has $\ell = 1$. Then zero time is required to reach a scope-1 configuration. However, this points out that our results may not hold for reaching a particular configuration. If, for example, $c_0 = \overline{0}a\overline{0}$ and we desire to reach $c = \overline{0}b\overline{0}$,

[1] $\lceil x \rceil$ denotes the smallest integer $\geq x$.

our results indicate that zero time is required but, in fact, at least one time step is needed. Thus, our results hold only for the scope of the configuration and not for the configuration itself. This statement will hold true for the remainder of this chapter.

6.3 Multi-dimensional Growth

We wish to extend our findings for scope-ℓ configurations in automata with scope-n neighborhoods to multi-dimensional automata. First, we must reconsider what is meant by "scope".

Definition 6.3.1 If and only if $\exists$ k, $k = (k_1, k_2, \ldots, k_d)$, $\bar{n}$, $\bar{n} = (n_1, n_2, \ldots, n_r, \ldots, n_d)$ where $\forall \ell$, $\ell = (\ell_1, \ell_2, \ldots, \ell_r, \ldots, \ell_d)$, $1 \leq \ell_r \leq n_r$, and $(k + \ell) \in X$, $\nexists \xi \neq (k + \ell)$, then X is called scope-$\bar{n}$.

An example of a scope-$\bar{n}$ neighborhood is in order. Consider E^2 TA with scope-(2,3) neighborhoods. One such neighborhood is X = ((0,0), (0,1), (0,2), (1,0), (1,1), (1,2)). For this neighborhood, k = (-1, -1). If sketched, X describes a rectangle of height two and width three. Another scope-(2,3) neighborhood in E^2 is X = ((1,1), (1,2), (1,3), (2,1), (2,2), (2,3)) which has k = (0,0). The d-tuple, k, is an offset.

Informally, a scope-$\bar{n}$ neighborhood index is a d-dimensional hyper-rectangle of length n_r in dimension r. Notice that neighborhoods which are not hyper-rectangles are not included. We also note that, since we have previously defined $X = (\xi_1, \xi_2, \ldots, \xi_n)$, then $|X| = n$ so a scope-$\bar{n}$ neighborhood has $n = \prod_{i=1}^{d} n_i$. By this, any cell in a tessellation automaton having a scope-$\bar{n}$ neighborhood still has n neighbors. Then our preliminary results of the previous section still hold.

Definition 6.3.2 If and only if $\exists i$, $i = (i_1, i_2, \ldots, i_d)$ where $\forall j$, $j = (j_1, j_2, \ldots, j_k, \ldots, j_d)$ and $0 \leq j_1 \leq \ell_1 - 1, \ldots, 0 \leq j_k \leq \ell_k - 1, \ldots, 0 \leq j_d \leq \ell_d - 1$, such that $c(i + j) \neq 0$ and $\forall p \neq i + j$, $c(p) = 0$ then c is said to be scope-$\bar{\ell}$, $\bar{\ell} = (\ell_1, \ell_2, \ldots, \ell_d)$ and is said to be located at position i.

Such a configuration can be thought of as a d-dimensional hyper-rectangle in which all cells within and on the boundaries are non-quiescent and those cells outside the hyper-rectangle are quiescent. The configuration has length ℓ_1 in dimension one, length ℓ_2 in dimension two and so forth. This differs from the definition for $\bar{\ell}(d)$ patterns of the previous chapter only in that the d-dimensional space is infinite. In short, the configuration, c, is simply an $\bar{\ell}(d)$ pattern embedded in an otherwise infinite quiescent tessellation and has a "corner" at location i. None of the symbols in the embedded $\bar{\ell}(d)$ pattern can be the quiescent symbol. We note that our definition does not include those configurations which are not hyper-rectangles or have quiescent cells embedded. We shall deal with these shortly. In essence, these definitions extend the notion of scope to be a d-tuple. Using this, we can extend our previous results to multi-dimensional automata.

Lemma 6.3.1 If X is scope-$\bar{n}$, c is scope-$\bar{r}$, σ is maximal with respect to c, then c' is scope-$[\bar{r} + \bar{n} - \bar{I}_d]$ where $\bar{I}_d = (1, 1, \ldots, 1)$ is the d-tuple of all ones.

Proof: Consider an arbitrary dimension, g, $1 \leq g \leq d$. In this dimension c has scope-r_g and X has scope-n_g. By an argument similar to that of

Lemma 6.2.5, we see that c' will have length $r_g + n_g - 1$ in dimension g. Since X is scope-$\bar{n}$, $\exists\ \xi_* \in X$, $\xi_* = ((k_1 + \ell_1), (k_2 + \ell_2), \ldots, (k_d + \ell_d))$. If c is at location i, then, clearly, the cell at location $i - \xi_*$ is initializable. Since X is scope-$\bar{n}$, then $\forall \xi \in X$, $\xi \neq \xi_*$, $\xi = (p_1, p_2, \ldots, p_r, \ldots, p_d)$, $\exists$ b, $1 \leq b \leq d$, $\ni p_b < (k_b + \ell_b)$. Then $\forall \xi$, $j = (j_1, j_2, \ldots, j_b, \ldots, j_d) = i - \xi$ the cell at location j is initializable but $\exists$ b, $1 \leq b \leq d$, $\ni j_b > i_b - (k_b + \ell_b)$. Then $i - \xi_*$ is the location of c'. Therefore, c' is scope-$[\bar{r} + \bar{n} - \bar{I}_d]$.

As in the case of one-dimensional automata, we can now find the configuration's scope after an arbitrary number of steps.

Lemma 6.3.2 If X is scope-$\bar{n}$, c_0 is scope-$\bar{r}$, σ is maximal with respect to $c_0, c_1, \ldots, c_{s-1}$, then c_s is scope-$[\bar{r} + s(\bar{n} - \bar{I}_d)]$.

Proof: The lemma is proven by an argument similar to Lemma 6.2.2.

Finally, we consider the minimum time to reach an arbitrary scope-$\bar{\ell}$ configuration.

Theorem 6.3.3 If X is scope-$\bar{n}$, c_0 is scope-$\bar{r}$, σ is maximal with respect to $c_0, c_1, \ldots, c_{t-1}$, then the minimum number of steps, t, to reach c, c is scope-$\bar{\ell}$, is $t = MAX\left|\frac{\bar{\ell}_i - \bar{r}_i}{n_i - 1}\right|$.

Proof: Consider dimension k, $1 \leq k \leq d$. At step s, the configuration has length $r_k + s(n_k - 1)$ in dimension k. Then, if $\forall k$, $s \geq \left|\frac{\bar{\ell}_k - \bar{r}_k}{n_k - 1}\right|$, then a configuration whose length is equal to or greater than $\bar{\ell}$ in every

dimension is reached. Therefore, $t = \left\lceil \frac{\ell_j - r_j}{n_j - 1} \right\rceil$ where $\forall i$, $t \geq \left\lceil \frac{\ell_i - r_i}{n_i - 1} \right\rceil$ is the minimum number of steps to reach c. More simply, this is $\text{MAX}\left\lceil \frac{\ell_i - r_i}{n_i - 1} \right\rceil = t$.

Corollary 6.3.4 If X is scope-$\bar{n}$, $c_0 = c_p$, σ is maximal with respect to $c_0, c_1, \ldots, c_{t-1}$, then the minimum number of steps, t, to reach c, c is scope-$\bar{\ell}$, is $t = \text{MAX}\left\lceil \frac{\ell_i - 1}{n_i - 1} \right\rceil$.

Proof: If $c_0 = c_p$, then c_0 is scope-(1, 1, ..., 1). Then $\bar{r}$ = (1, 1, ..., 1) so $\forall j$, $r_j = 1$. Thus, by Theorem 6.3.3, $t = \text{MAX}\left\lceil \frac{\ell_i - 1}{n_i - 1} \right\rceil$.

As we stated previously, these results hold only for prediction of the configuration's scope and in no way allow us to predict the configuration itself. This concludes our investigation of growth of scope-$\bar{\ell}$ configurations in tessellation automata having scope-$\bar{n}$ neighborhoods.

6.4 Irregular Neighborhoods and Configurations

Our efforts of the previous sections confined their attention to configurations and neighborhoods which were contiguous. It is exactly this constraint which will be of greatest value in our next chapter. However, some results may be obtained for automata in which this is not the case. In these automata, prediction of any property other than the size of the boundaries of the configuration is not possible due to the fact that the neighborhood is not contiguous and that the initial configuration may not have all cells initialized within its perimeter.

Definition 6.4.1 A configuration, c, is said to be span-$\bar{\ell}$, if and only if $\exists$ a scope-$\bar{\ell}$ configuration, g, where $\forall i \ni c(i) \neq 0$, then $g(i) \neq 0$ and $\nexists$ scope-$\bar{m}$ configuration, h, $\ni \exists\ m_r < \ell_r$, $1 \leq r \leq d$, where $\forall i \ni c(i) \neq 0$ then $g(i) \neq 0$.

We are simply finding the smallest scope-$\bar{\ell}$ configuration, g, in which the configuration, c, could be placed. In other words, c and g occupy the same tessellation space but not every cell in c is necessarily initialized. Note that this definition is completely general. That is, any configuration may be classed as span-$\bar{\ell}$, for some $\bar{\ell}$.

Definition 6.4.2 A neighborhood index, X, is said to be span-$\bar{n}$ if and only if $\exists$ a neighborhood, Y, Y is scope-$\bar{n}$, where $\forall\ \xi \in X$, then $\xi \in Y$ and $\nexists$ scope-$\bar{m}$ neighborhood, Z, $m_r < n_r$, $\ni \forall \xi \in X$, $\xi \in Z$.

A neighborhood which is span-$\bar{n}$ has all its elements in some minimal scope-$\bar{n}$ neighborhood. We note that any neighborhood index may be classed as a span-$\bar{n}$ neighborhood, for some $\bar{n}$.

Lemma 6.4.1 If X is span-$\bar{n}$, c is span-$\bar{r}$, σ is maximal with respect to c, then c' is span-$(\bar{r} + \bar{n} - \bar{I}_d)$.

Proof: Consider the configuration, c, and an arbitrary dimension, k, $1 \leq k \leq d$. Since c is scope-$\bar{\ell}$, $\exists$ i, j $\ni$ $c(i) \neq 0$, $c(j) \neq 0$, $i_k - j_k = r_k - 1$. That is, there are two cells which are a distance of $r_k - 1$ from each other in dimension k. Since X is span-$\bar{n}$, $\exists\ \xi_e, \xi_f \in X$, $\xi_e = (z_{1e}, z_{2e}, \ldots, z_{de})$, $\xi_f = (z_{1f}, z_{2f}, \ldots, z_{df}) \ni z_{ke} - z_{kf} = n_k - 1$. Then the cell at $i - \xi_e$ has $c(N(X, i - \xi_e)) \neq 0$ and is initializable.

Also, the cell at $j - \xi_f$ is initializable. The distance between these two cells in dimension k is $(j_k - z_{kf}) - (i_k - z_e) = j_k - z_{kf} - i_k + z_{ke}$ $= (j_k - i_k) + (z_{ke} - z_{kf}) = r_k + n_k - 2$. Since σ is maximal with respect to c, these cells will be initialized. Thus, the span of c' in dimension k is $r_k + n_k - 2 + 1 = r_k + n_k - 1$. Since this is the case for all k, then c' is span-$(\bar{r} + \bar{n} - \bar{I}_d)$.

As noted earlier, since a span-ℓ configuration need not have all cells initialized, then the next configuration, too, may contain blocks of quiescent cells. Under these conditions, only the growth of the boundary can be predicted. We state our final results for this case.

Lemma 6.4.2 If X is span-$\bar{n}$, c_0 is span-$\bar{r}$, σ is maximal with respect to $c_0, c_1, \ldots, c_{s-1}$, then c_s is span-$[\bar{r} + s(\bar{n} - \bar{I}_d)]$.

Proof: The lemma is proven by an argument similar to Lemma 6.2.2.

Theorem 6.4.3 If X is span-$\bar{n}$, c_0 is span-$\bar{r}$, σ is maximal with respect to $c_0, c_1, \ldots, c_{t-1}$, then the minimum number of steps, t, to reach c, c is span-$\bar{\ell}$, is $t = \text{MAX}\left\lceil \frac{\ell_i - r_i}{n_i - 1} \right\rceil$.

Proof: The Theorem is proven by an argument similar to Theorem 6.3.3.

Corollary 6.4.4 If X is span-$\bar{n}$, $c_0 = c_p$, σ is maximal with respect to $c_0, c_1, \ldots, c_{t-1}$, then the minimum number of steps, t, to reach c, c is span-$\bar{\ell}$, is $t = \text{MAX}\left\lceil \frac{\ell_i - 1}{n_i - 1} \right\rceil$.

This completes our findings for growth rate in span-$\bar{\ell}$ configurations. These configurations will be of little interest within our application

due to the fact that each configuration during the growth process may not have all cells initialized. This difficulty arises from the fact that the neighborhood index is not contiguous.

Definition 6.4.3 A span-$\bar{n}$ neighborhood index, X, is said to have inner scope-$\bar{m}$, (Iscope-$\bar{m}$) if and only if $\exists$ a scope-$\bar{m}$ neighborhood index, Y, $\forall\ \xi \in Y$, then $\xi \in X$ and $\nexists$ scope-$\bar{p}$ neighborhood index, Z, where $\exists\ r \ni p_r > m_r$, $\forall\ \xi \in Z$, $\xi \in X$.

Then, the Iscope of a span-$\bar{n}$ neighborhood index is the largest scope-$\bar{m}$ index contained in it. We wish to employ the notion of Iscope with respect to configurations as well.

Definition 6.4.4 A span-$\bar{\ell}$ configuration, c, is said to have Iscope-$\bar{p}$ if and only if $\exists$ a scope-$\bar{p}$ configuration, d, $\ni \forall i$, $d(i) \neq 0$, then $c(i) \neq 0$ and $\nexists$ scope-$\bar{q}$ configuration, e, $\exists\ q_r > p_r$, $\ni \forall i$, $e(i) \neq 0$ then $c(i) \neq 0$.

This definition is consistent with the notion of Iscope for neighborhoods. The Iscope of a configuration is the largest scope-$\bar{p}$ configuration found within it. Then, a span-$\bar{\ell}$ configuration (or neighborhood) which is also Iscope-$\bar{\ell}$ is really a scope-$\bar{\ell}$ configuration (or neighborhood). We also point out that any configuration or neighborhood has some Iscope even if it is simply (1, 1, ..., 1).

Lemma 6.4.5 If X is span-$\bar{n}$ Iscope-$\bar{m}$, c is span-$\bar{r}$ Iscope-$\bar{p}$, σ is maximal with respect to c, then c' is span-$(\bar{r} + \bar{n} - \bar{I}_d)$ Iscope-$\bar{q}$, for some $\bar{q}$ where $\forall i$, $1 \leq i \leq d$, $q_i \geq p_r + m_r - 1$.

Proof: Lemma 6.4.1 demonstrates that c' is span-$(\bar{r} + \bar{n} - \bar{I}_d)$. If we

consider Lemma 6.3.1, then clearly c' has Iscope of at least $(\bar{p} + \bar{m} - \bar{I}_d)$.

The Lemma is somewhat unsettling in that the Iscope of c' is not predicted exactly. We have merely found its minimal size. We will demonstrate by two examples that it may be larger than the minimum of $\bar{p} + \bar{m} - \bar{I}_d$ if either X is not scope-$\bar{n}$ or c is not scope-$\bar{p}$. For simplicity, we shall consider each occurrence of "a" to be an arbitrary symbol in A.

Example 6.4.1 Consider an E^1 tessellation automaton having X = (0, 1, 2) and c = $\bar{0}$aa0aa$\bar{0}$. Then X is scope-3 and c is span-5 Iscope-2. If σ is maximal with respect to c, then c' = $\bar{0}$aaaaaaa$\bar{0}$ which is scope-7. The predicted lower bound for Iscope of c' is 2 + 3 - 1 = 4.

Example 6.4.2 Consider an E^1 tessellation automaton having X = (-2, 0, 1) and c = $\bar{0}$aa$\bar{0}$. Then X is span-4 Iscope-2 and c is scope-2. If σ is maximal with respect to c, then c' = $\bar{0}$aaaaa$\bar{0}$ which is scope-5. The predicted lower bound for the Iscope of c' is 2 + 2 - 1 = 3.

The two examples point out the unpredictable nature of growth when arbitrary span-$\bar{n}$ neighborhoods or configurations are permitted. We can only find upper and lower bounds on the growth rate using this result.

Lemma 6.4.6 If X is span-$\bar{n}$ Iscope-$\bar{m}$, c_0 is span-$\bar{r}$ Iscope-$\bar{p}$, σ is maximal with respect to the sequence $c_0, c_1, \ldots, c_{s-1}$, then c_s is span-$[\bar{r} + s(\bar{n} - \bar{I}_d)]$ scope-$\bar{q}$ for some $\bar{q}$ where $\forall i$, $1 \leq i \leq d$, $q_i \geq p_r + m_r - 1$.

Theorem 6.4.7 If X is span-$\bar{n}$ Iscope-$\bar{m}$, c_0 is span-$\bar{r}$, Iscope-$\bar{p}$, σ is maximal with respect to the sequence $c_0, c_1, \ldots, c_{t-1}$ then to reach c, c

is scope-$\bar{\ell}$, t time steps are required where $t_a \leq t \leq t_b$, $t_a = \text{MAX}\left|\frac{\ell_i - r_i}{n_i - 1}\right|$, $t_b = \text{MAX}\left|\frac{\ell_i - p_i}{m_i - 1}\right|$.

Proof: Since the span of c_t must be at least equal to the span of c in each dimension, we employ Theorem 6.4.3 to find the lower bound, t_a. By considering only the Iscope-$\bar{p}$ portion of c_0, then by Theorem 6.3.3, we are assured of achieving a scope-$\bar{\ell}$ configuration at t_b.

Corollary 6.4.8 If X is span-$\bar{n}$ Iscope-$\bar{m}$, $c_0 = c_p$, σ is maximal with respect to $c_0, c_1, \ldots, c_{t-1}$, then to reach c, c is scope-$\bar{\ell}$, t time steps are required where $t_a \leq t \leq t_b$, $t_a = \text{MAX}\left|\frac{\ell_i - 1}{n_i - 1}\right|$, $t_b = \text{MAX}\left|\frac{\ell_i - 1}{m_i - 1}\right|$.

The Corollary particularizes Theorem 6.4.7 to the case where the initial configuration is primitive. In this case, c_0 is span-(1, 1, ..., 1) Iscope-(1, 1, ..., 1). This concludes our findings for growth rate in tessellation automata.

6.5 Discussion

In this chapter, we have found the relationship between the rate of growth of a configuration and the tessellation automaton's neighborhood index. Theorem 6.4.7, which is the main result for completely arbitrary configuration and neighborhood index, simply finds upper and lower bounds for the time required to reach a configuration having a contiguous block of initialized cells. Within this dissertation, the results of Theorem 6.3.3, which finds the time to reach a configuration in which all cells of a given sized block are initialized where both the initial configura-

tion and the neighborhood index are themselves contiguous, will be of greater value.

VII. SYNTHESIS

7.1 Introduction

In Chapter III, it was proposed that the tessellation automaton govern the global function of the array. As was stated, the function of each cell in the computing array is determined from the state of the associated cell in the controlling array (tessellation automaton). Then, for some given global configuration, c, of the tessellation automaton, there is a corresponding arrangement of functional blocks in the computing array. This defines the global function of the computing array. Consider the mapping, h, h: $A \rightarrow F$. Certainly, we require that $|A| \geq |F|$ or there are functions in the cells of the computing array which cannot be invoked by the tessellation automaton. We wish to partition the alphabet of the tessellation automaton, A, such that $A = A_s \cup A_f$, and specify that $|A_f| \geq |F|$ and $\forall a \in A_f$, $\exists$ a unique $f \in F$ where $h(a) = f$. This partially specifies the mapping h which will then be "onto" between A_f and F. We see that under these conditions, a desired arrangement of functions in the computing array has an associated global configuration which the tessellation automaton must assume if the desired arrangement is to be invoked. We also note that it is made up entirely from symbols in A_f. The symbols in A_s will be employed in the synthesis of the desired global configuration.

Clearly, the global configuration which invokes the desired computing function should be in equilibrium for if it is not, the tessellation automaton will alter its global configuration at the next time step. This implies that the function of the computing array will also change and the overall machine will not be performing the correct function.

With this in mind, we recall Theorem 4.4.5 which states:

$\forall c \in C, \exists\ \sigma\ ^{3}\ c$ is an equilibrium configuration if $\xi_o \in X$.

We also recall Theorem 4.4.6.

If $\xi_o \notin X, \exists\ \sigma\ ^{3}\ c$ is an equilibrium configuration if and only if $\forall i,j\ ^{3}\ c(N(X,i)) = c(N(X,j))$, then $c(i) = c(j)$.

Then, if $\xi_o \notin X$, there are some configurations which cannot be in equilibrium in the tessellation automaton. For each of these configurations, there is some computing function which cannot be realized. Since we cannot be certain that these functions are not valuable, we should not rule out their potential utilization. For this reason, we resolve that in the tessellation automaton, $\xi_o \in X$.

Now, we consider how the tessellation automaton is to acquire the desired configuration. While it is possible to initialize each and every cell "by hand", such a procedure would be time consuming at best. Furthermore, it is possible, or even likely, that an incorrect configuration might be "loaded" into the automaton. In fact, if this method is employed, the tessellation automaton could be replaced with a simple programmable memory. Obviously, that cannot be permitted! We propose that the desired configuration be achieved by placing the automaton into a particular initial configuration and then allowing it to move sequentially through several intermediate configurations until the desired final configuration is reached. Since ξ_o is presumed to be in X, we are assured that σ exists so that the desired configuration is, in fact, a final (equilibrium) configuration. We now define the formal notion of

synthesis.

Definition 7.1.1 If and only if $\tau^s(c_o) = c_f$ and $\tau(c_f) = c_f$, then c_f is said to be synthesized from c_o in s steps.

Definition 7.1.2 If and only if c_f can be synthesized from c_o, then c_o is said to be a seed of c_f.

We shall make some restrictions on X and c_o. So that the growth rate of the automaton's configuration be predictable, we will require that c_o be a scope-$\bar{r}$ configuration and that X be scope-$\bar{n}$. Then, by Lemma 6.3.1, each succeeding configuration can be contiguous if σ is maximal at each step and we can employ Theorem 6.3.3 to determine the minimal number of steps to reach a contiguous configuration which can be mapped to c_f. We will not require that c_f be contiguous since it is to be an arbitrary configuration. It will be considered as a span-$\bar{\ell}$ configuration and no attempt to determine its Iscope is required. Since the fundamental advantage of utilizing synthesis is to avoid entering a large configuration, we will assume that if c_o is scope-$\bar{r}$ and c_f is span-$\bar{\ell}$, then $\forall i$, $r_i \leq \ell_i$, and $\exists\ j \ni r_j < \ell_j$. In other words, we require that c_o be smaller than c_f in some dimension and that it be no larger in any dimension. Certainly, if c_o were a primitive configuration, we have the minimal amount of information to enter. We shall not restrict c_o to be a primitive configuration for two reasons. Gregory [9] correctly reasons that if a fault exists in the cell utilized for the non-quiescent symbol of the primitive configuration, synthesis may be doomed. Such a disaster might be avoided if a larger initial configuration is employed. Secondly,

use of primitive configurations may not be practical if a large set of final configurations are to be permitted. This will be discussed in a later section of this chapter.

We must consider the existence of σ. Inherent in our definition of synthesis is the implication that the tessellation automaton is stable. We recall that this question was shown unsolvable in Chapter IV for arbitrary TA in an arbitrary configuration. We should realize that the question of stability might be resolved for particular tessellation automata in a particular initial configuration. The non-erasing automata mentioned in Chapter IV illustrate such a possibility. These automata were shown to be unconditionally stable. Fortunately, the question of stability is partially resolved. Since we require that $\xi_o \in X$, then we are assured that σ exists so that c_f is an equilibrium configuration. If we demonstrate that c_f is reached at some point in time, then stability is guaranteed. We now demonstrate, by example, that this question is not trivial.

Example 7.1.1

Synthesize the configuration $\overline{0}\ 1\ 2\ 3\ \overline{0}$ in $(\{0,1,2,3\}, E^1, (-1,0),\sigma)$. That is, define σ (if possible). Let $c_o = \overline{0}\ 1\ \overline{0}$ and σ:

$00 \to 0$	$20 \to 3$
$01 \to 1$	$21 \to -$
$02 \to -$	$22 \to -$
$03 \to -$	$23 \to 3$
$10 \to 2$	$30 \to 0$
$11 \to -$	$31 \to -$
$12 \to 2$	$32 \to -$
$13 \to -$	$33 \to -$

("-" denotes an unspecified symbol)

Using this σ, the following array transitions occur.

$$\overline{0}\ 1\ \overline{0}$$

$$\overline{0}\ 1\ 2\ \overline{0}$$

$$\overline{0}\ 1\ 2\ 3\ \overline{0}$$

$$\overline{0}\ 1\ 2\ 3\ \overline{0}$$

etc.

Notice that the synthesis terminates in an equilibrium configuration. That is, no further changes occur. Therefore, this automaton is stable (under the specified σ) and, as shown, synthesizes $c_f = \overline{0}\ 1\ 2\ 3\ \overline{0}$. The existence of σ to perform the synthesis for any specified configuration must be demonstrated as shown by the next example.

Example 7.1.2

The configuration $\overline{0}\ 1\ 0\ 1\ 1\ 1\ \overline{0}$ cannot be synthesized in $(\{0,1\}, E^1, (-1,0,1), \sigma)$. For equilibrium, we require,

$$\begin{aligned}
\sigma:\ & 0\ 0\ 0 \to 0\\
& 0\ 0\ 1 \to 0\\
& 0\ 1\ 0 \to 1\\
& 0\ 1\ 1 \to 1\\
& 1\ 0\ 0 \to 0\\
& 1\ 0\ 1 \to 0\\
& 1\ 1\ 0 \to 1\\
& 1\ 1\ 1 \to 1
\end{aligned}$$

Since all neighborhood configurations are specified, we cannot ensure the synthesis using σ. Only two seeds exist, 0 and 1, and therefore only

$\overline{0}\ 1\ \overline{0}$ or $\overline{1}\ 0\ \overline{1}$ can be used for c_o. Trying these yields:

$$\begin{array}{ll} \overline{0}\ 000\ 1\ 000\ \overline{0} & \overline{1}\ 111\ 0\ 111\ \overline{0} \\ \overline{0}\ 000\ 1\ 000\ \overline{0} & \overline{1}\ 111\ 0\ 111\ \overline{0} \\ \vdots & \vdots \end{array}$$

Thus, the configuration cannot be synthesized in the specified automaton. Certainly, if X or A were larger, synthesis might be possible. This is the main question to be considered in this chapter. That is, given an automaton and a desired equilibrium configuration, how many symbols in A are required to guarantee the existence of σ whereby the desired configuration is synthesized from a simple initial configuration.

The informed reader who is familiar with the work of Yamada-Amoroso [29] may wonder how synthesis differs from the completeness problem. Completeness considers the question of whether or not there exists a configuration which cannot evolve from a primitive configuration in a given tessellation automaton. The automaton under consideration allows its local transformation, σ, to vary in time. Then completeness implies that for every configuration, there exists a sequence of local transformations which, if applied to a primitive configuration, yields the given configuration. This question is tightly coupled to Moore's Garden-of-Eden problem [16] since, if an automaton is complete, there exists no Garden-of-Eden configurations. Synthesis and completeness differ in three fundamental ways. First, the tessellation automata which we are considering has a single (selected) local transformation and is not allowed to vary in time. Second, we have no qualms about increasing the

size of the automaton's alphabet, A, in order to reach a given configuration. Third, the basic question is somewhat different in that we are concerned with the evolution of a particular configuration, not all configurations.

7.2 Synthesis in One-dimensional Automata

In this section, we consider the synthesis problem for one-dimensional tessellation automata. In accordance with our introduction, we require that X be scope-n and that c_o be scope-r. We also require that c_f be span-ℓ and that $r < \ell$. Furthermore, we will assume that c_f is made up from symbols in A_f. We can find the required size of A_s which assures the existence of σ so that any given c_f is synthesized. Our initial result considers the case where c_o is a primitive configuration.

Theorem 7.2.1 In E^1 TA, if X is scope-n, $\xi_o \in X$, c_f is span-ℓ, $c_o = c_p$, $|A_s| \geq 2 + \sum_{s=1}^{t-1} \left\lceil \sqrt[n]{1+(s-1)(n-1)} \right\rceil$, where $t = \left\lceil \frac{\ell - 1}{n - 1} \right\rceil$, then $\exists\, \sigma \ni c_f$ is synthesized from c_o.

Proof: We will require that each new configuration, c', that arises during synthesis be scope-(r+n-1) where the previous configuration was scope-r. Then clearly, σ, if it exists, will be maximal at each step. Then, by Lemma 6.2.6, the configuration at the s^{th} step will be scope-[1 + s(n-1)] and by Corollary 6.2.8, a scope-ℓ configuration will be reached at $t = \left\lceil \frac{\ell - 1}{n - 1} \right\rceil$. To insure the existence of σ, we will construct a configuration at each step using a unique set of symbols and having all c(N(X,i)) unique. This can be accomplished by using (k^n+n-1, n) strings embedded in a quiescent tessellation at each step, s. If the

quiescent symbol, 0, is not used in the (k^n+n-1, n) strings, then the $c(N(X,i))$ that border (contain quiescent cells that are initializable) the string are also unique due to the varying number of quiescent symbols in each neighborhood. It is not necessary to use a (k^n+n-1, n) string at step t since at t - 1 the configuration can give rise to a span-ℓ configuration at step t. Since, at t - 1, all $c(N(X,i))$ are unique, we can choose σ to yield c_f at step t. Then, we simply count the number of symbols required. At step s, we require a scope-$(1 + s(n-1))$ configuration. By Theorem 5.3.5, we need k_s symbols, $k_s = \left\lceil \sqrt[n]{[1 + s(n-1)] - n + 1} \right\rceil = \left\lceil \sqrt[n]{1 + (s-1)(n-1)} \right\rceil$ at step s, $s > 0$. At $s = 0$, $c_o = c_p$ so we require one symbol for the primitive and the quiescent symbol,0 . Summing these yields:

$$2 + \sum_{s=1}^{t-1} \left\lceil \sqrt[n]{1 + (s-1)(n-1)} \right\rceil$$

where $t = \left\lceil \frac{\ell - 1}{n - 1} \right\rceil$. This is the required cardinality of A_s. Thus,

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} \left\lceil \sqrt[n]{1 + (s-1)(n-1)} \right\rceil$$

assures the existence of σ to reach c_f. Since $\xi_o \in X$, we are assured by Theorem 4.4.5 that σ exists such that c_f is an equilibrium configuration.

The following example demonstrates the theorem.

Example 7.2.1 Synthesize the configuration $\bar{0}$MICHELLE$\bar{0}$ in $(A, \acute{E}^1, (-1,0,1), \sigma)$. Since $c_f = \bar{0}$MICHELLE$\bar{0}$, we require $A_f = \{C, E, H, I, L, M\}$ and, since X = (-1,0,1), then n = 3. We see that if $c_o = c_p = \bar{0}\ A\ \bar{0}$ then c_1 will be scope-3. We will employ a (3,3) string for c_1. Such a string

is BBB so let $c_1 = \overline{0}BBB\overline{0}$. Continuing, we see that c_2 will be scope-5 so we will employ a (5,3) string for c_2. Such a string is FGGFF so let $c_2 = \overline{0}FGGF\overline{F0}$. At the next step, c_3 will be scope-7, which requires a (7,3) string. The string SSSTSTT is a (7,3) string so we let $c_3 = \overline{0}SSSTSTT\overline{0}$. Then c_4 will be scope-9 which permits $c_4 = c_f$. Therefore, we shall use $A_s = \{A, B, F, G, S, T\}$ and define σ as:

c(N(X,i))	σ(c(N(X,i))
1 - 000	0
00A	B
2 - 0A0	B
A00	B
00B	F
0BB	G
3 - BBB	G
BB0	F
B00	F
00F	S
0FG	S
FGG	S
4 - GGF	T
GFF	S
FF0	T
F00	T
00S	M
0SS	I
SSS	C
SST	H
5 - STS	E
TST	L
STT	L
TT0	E
T00	0
00M	0
0MI	M
MIC	I
ICH	C
CHE	H
6 - HEL	E
ELL	L
LLE	L
LE0	E
E00	0

Portion 1 of σ is required to prevent cells in a quiescent neighborhood

configuration from changing state. Portion 2 causes the initial primitive configuration $c_o = \overline{0}A\overline{0}$ to be mapped (in τ) to $c_1 = \overline{0}BBB\overline{0}$. Portion 3 causes $c_2 = \overline{0}FGGFF\overline{0}$ and portion 4 generates $c_3 = \overline{0}SSSTSTT\overline{0}$. Portion 5 causes $\overline{0}$MICHELLE$\overline{0}$ to appear as c_4 and portion 6 assures that c_4 is an equilibrium configuration.

Using σ, the following array transitions occur.

---000000A000000---

---00000BBB00000---

---0000FGGFF0000---

---000SSSTSTT000---

---00MICHELLE000---

---00MICHELLE000---

etc.

This is no surprise since each intermediate configuration was selected in advance. We are assured that σ exists because every neighborhood configuration during the synthesis steps is unique. We note that MICHELLE itself has all neighborhood configurations unique. This condition is not required since c_f need only be an equilibrium configuration and $\xi_o \in X$ assures us that σ exists so that it is.

It should be clear that any arbitrary span-ℓ configuration, $\ell \leq 9$, could have been synthesized by modifying portions 5 and 6 of the local transformation. If $\ell \leq 7$, A_s could be made smaller. The bound on A_s found in Theorem 7.2.1 is:

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} \left\lceil \sqrt[3]{1 + (s-1)(3-1)} \right\rceil = 7, \quad t = \left\lceil \frac{8-1}{3-1} \right\rceil = 4$$

which is exactly the number of symbols employed for the example.

Table 7.2.1 shows the number of steps required and the symbols needed to reach configurations of various lengths for several different scope-n neighborhoods.

Next, we consider growth from initial configurations which are not primitive. We will require c_o to be a scope-r configuration, which is an (r,n) string embedded in a quiescent tessellation. This is required in order that c_1 be chosen without regard for c_o and still be certain that σ exists. If all c(N(X,i)) are unique, then no difficulty arises. This same requirement exists for the initial configuration when primitives are employed; however, any primitive configuration satisfies the condition.

Corollary 7.2.2 In E^1 TA, if X is scope-n, $\xi_o \in X$, c_f is span-ℓ, c_o is scope-r, $n \leq r < \ell$, $|A_s| \geq 1 + \sum_{s=0}^{t-1} \left\lceil \sqrt[n]{r + (s-1)(n-1)} \right\rceil$, where $t = \left\lceil \frac{\ell - r}{n - 1} \right\rceil$, then $\exists\, \sigma \ni c_f$ is synthesized from c_o.

Proof: By Theorem 6.2.7, a scope-ℓ configuration can be reached at $t = \left\lceil \frac{\ell - r}{n - 1} \right\rceil$. We require the quiescent symbol plus enough symbols to write (r + s(n-1), n) strings at each step until t - 1. Then,

$$|A_s| \geq 1 + \sum_{s=1}^{t-1} \left\lceil \sqrt[n]{[r + s(n-1)] - n + 1} \right\rceil$$

$$= 1 + \sum_{s=0}^{t-1} \left\lceil \sqrt[n]{r + (s-1)(n-1)} \right\rceil$$

is sufficient.

Our next result for strings considers synthesis from initial config-

Table 7.2.1 Number of symbols and times to synthesize configurations of length ℓ in scope-n E^1 automata

		n				
	k	2	3	4	5	6
ℓ	5	7	5	4	4	3
	10	12	7	6	5	4
	15	17	10	7	6	5
	20	22	12	9	7	6

		n				
	t	2	3	4	5	6
ℓ	5	5	3	2	2	1
	10	10	5	4	3	2
	15	15	8	5	4	3
	20	20	10	7	5	4

urations that are scope-r, $1 \leq r \leq n$.

Corollary 7.2.3 In E^1 TA, if X is scope-n, $\xi_o \in X$, c_f is span-ℓ, c_o is scope-r, $1 \leq r \leq n$,

$$|A_s| = 2 + \sum_{s=1}^{t-1} \left\lceil \sqrt[n]{r + (s-1)(n-1)} \right\rceil,$$

where $t = \left\lceil \frac{\ell - r}{n - 1} \right\rceil$, then $\exists\, \sigma \ni c_f$ is synthesized from c_o.

Proof: Since c_o is scope-r, $1 \leq r \leq n$, we consider the number of symbols required so that c_o has all unique $c(N(X,i))$. We see that $\overline{oa^{r-}0}$ satisfies this criteria because $n \geq r$. This is due to the varying number of quiescent symbols in each neighborhood configuration. Then two symbols are required, which is the same number required for a primitive configuration. Thus, the results of Theorem 7.2.1 are valid if $1 \leq r \leq n$.

The bound on $|A_s|$ found in these results is somewhat unwieldly. We can find a bound somewhat greater than those found previously but which is a simpler result. The bound on $|A_s|$ for synthesis from primitive configurations is worst case since if c_o is scope-r, $r > n$, then the steps required to reach a scope-r configuration from a primitive configuration are not needed. The deletion of these steps is accompanied by a reduction in $|A_s|$. Thus, we will concentrate on synthesis from a primitive configuration to find a new bound. Rearranging the bound of Theorem 7.2.1, we have:

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} \left\lceil \sqrt[n]{1 + (s-1)(n-1)} \right\rceil, \quad t = \left\lceil \frac{\ell - 1}{n - 1} \right\rceil.$$

We consider the function $1 + (s-1)(n-1)$ and claim $\forall\, s > 0$, $n > 0$, then

$s^n \geq 1 + (s-1)(n-1)$. Since:

$$\frac{d}{ds}\left[s^n\right] = n\, s^{n-1}$$

$$\frac{d}{ds}\left[1 + (s-1)(n-1)\right] = n - 1$$

then $\frac{d}{ds}\left[s^n\right] > \frac{d}{ds}\left[1 + (s-1)(n-1)\right]$ if $s > 0$, $n > 0$. At $s = 1$, then:

$$s^n\Big|_{s=1} = 1,\ 1 + (s-1)(n-1)\Big|_{s=1} = 1$$

the functions s^n, $1 + (s-1)(n-1)$, are equal. Since we have equality at $s = 1$ and since $\frac{d}{ds}\left[s^n\right] > \frac{d}{ds}\left[1 + (s-1)(n-1)\right]$, we may conclude that $s^n \geq 1 + (s-1)(n-1)$. Using this in our bound on $|A_s|$ yields:

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} \left\lceil \sqrt[n]{s^n} \right\rceil$$

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} s$$

$$|A_s| \geq 2 + \frac{t(t-1)}{2}$$

$$|A_s| \geq \frac{t^2-t+4}{2}$$

$$\text{where } t = \left\lceil \frac{\ell - 1}{n - 1} \right\rceil.$$

Thus, we have shown:

Corollary 7.2.4 In E^1 TA, if X is scope-n, $\xi_o \in X$, c_f is span-ℓ, c_o is scope-r, $|A_s| \geq \frac{t^2-t+4}{2}$, $t = \left\lceil \frac{\ell - 1}{n - 1} \right\rceil$, then $\exists\, \sigma\ni$ c_f is synthesized from

c_o in t or fewer steps.

Although this bound may be considerably larger than those found previously, it is much easier to compute. If the automaton is to be implemented, the previous bounds should be employed since every symbol in the automaton's alphabet is a state which must be included in each cell's realization. Fewer states in the cell, due to a reduction in alphabet cardinality, will probably yield a simpler cell. However, since any extra states will have unspecified transitions, this may not be the case. The cell designer could utilize this fact to produce a desired realization.

We demonstrate similar results for automata with span-p, Iscope-n neighborhoods.

Corollary 7.2.5 In E^1 TA, if X is Iscope-n, $\xi_o \in X$, c_f is span-ℓ, c_o is scope-r, $1 \leq r \leq n$, $|A_s| \geq 2 + \sum_{s=1}^{t-1} \left\lceil \sqrt[n]{r + (s-1)(n-1)} \right\rceil$, where $t = \left\lceil \frac{\ell - r}{n - 1} \right\rceil$, then $\exists\, \sigma^3\, c_f$ is synthesized from c_o.

The corollary is proven trivially by considering only the Iscope-n portion of X. The extra elements in X do not prevent the configurations from having unique neighborhood configurations even if ξ_o is not in the Iscope portion of X. The presence of ξ_o in X is still required to ensure equilibrium of c_f.

Corollary 7.2.6 In E^1 TA, if X is Iscope-n, $\xi_o \in X$, c_f is span-ℓ, c_o is scope-r, $n \leq r < \ell$, $|A_s| \geq 1 + \sum_{s=0}^{t-1} \left\lceil \sqrt[n]{r + (s-1)(n-1)} \right\rceil$ where $t = \left\lceil \frac{\ell - r}{n - 1} \right\rceil$ then $\exists\, \sigma^3\, c_f$ is synthesized from c_o.

Corollary 7.2.7 In E^1 TA, if X is Iscope-n, $\xi_o \in X$, c_f is span-ℓ, c_o is scope-r, $|A_s| \geq \frac{t^2-t+4}{2}$, $t = \left\lceil \frac{\ell - 1}{n - 1} \right\rceil$, then $\exists \sigma \ni c_f$ is synthesized from c_o in t or fewer steps.

This concludes our study of synthesis in one-dimensional tessellation automata. Corollary 7.2.7 is our most general result which permits a simple calculation of the bound on A_s which ensures synthesis for arbitrary neighborhood index.

7.3 Synthesis in Multi-dimensional Automata

We wish to present results similar to those of the previous section which relate to multi-dimensional automata. Initially, we consider synthesis of span-$\bar{\ell}$ configurations from a primitive configuration in automata having scope-$\bar{n}$ neighborhoods containing ξ_o. Then, corollaries are presented which cover the cases for non-primitive initial configuration.

When considering synthesis in multidimensional automata, the problem of embedding $(\ell,n)_d$ patterns in a quiescent tessellation must be dealt with. Unfortunately, when these patterns are embedded, not all c(N(X,i)) are unique. The following configuration is a $(10,3)_2$ pattern embedded in a quiescent tessellation.

$$
\begin{array}{ccc}
 & \vdots & \\
 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & \\
 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & \\
 & 0\ 0\ A\ B\ B\ B\ A\ B\ A\ A\ A\ B\ 0\ 0 & \\
 & 0\ 0\ A\ B\ B\ B\ A\ B\ A\ A\ A\ B\ 0\ 0 & \\
 & 0\ 0\ A\ B\ B\ B\ A\ B\ A\ A\ A\ B\ 0\ 0 & \\
 & 0\ 0\ B\ B\ B\ A\ B\ A\ A\ A\ B\ B\ 0\ 0 & \\
 & 0\ 0\ B\ A\ B\ A\ A\ A\ B\ B\ B\ \boxed{A\ 0\ 0} & \\
\cdots & 0\ 0\ A\ A\ A\ B\ B\ B\ A\ B\ A\ \boxed{A\ 0\ 0} & \cdots \\
 & 0\ 0\ B\ B\ A\ B\ A\ A\ A\ B\ B\ \boxed{B\ 0\ 0} & \\
 & 0\ 0\ A\ A\ B\ B\ B\ A\ B\ A\ A\ \boxed{A\ 0\ 0} & \\
 & 0\ 0\ B\ A\ A\ A\ B\ B\ B\ A\ B\ \boxed{A\ 0\ 0} & \\
 & 0\ 0\ A\ B\ A\ A\ A\ B\ B\ B\ A\ \boxed{B\ 0\ 0} & \\
 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & \\
 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & \\
 & \vdots &
\end{array}
$$

The pattern is the $(10,3)_2$ pattern found in Section 5.5 where "0" is replaced by "A" and "1" is replaced by "B". When the pattern is embedded, there are subpatterns along the boundaries which are not unique. The following theorem defines a construction which will alleviate this difficulty.

Theorem 7.3.1 In E^d TA, if X is scope-$\overline{n}$, $\xi_o \in X$, c_f is span-$\overline{\ell}$, $c_o = c_p$,

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} \left(\left\lceil \overline{MIN_n} \sqrt{sMAX_n - MIN_n - s} \right\rceil + \left\lceil \overline{MIN_n} - 1 \sqrt{sMAX_n - MIN_n - s+2} \right\rceil \right),$$

where $t = MAX \left\lceil \frac{\overline{\ell_i} - 1}{n_i - 1} \right\rceil$ then $\exists\ \sigma \ni c_f$ is synthesized from c_o.

Proof: By Corollary 6.3.4, we can reach a scope-$\overline{\ell}$ configuration at $t = \left\lceil \frac{\overline{\ell_j} - 1}{n_j - 1} \right\rceil$ where $\forall i$, $t \geq \left\lceil \frac{\overline{\ell_i} - 1}{n_i - 1} \right\rceil$. At the s^{th} step, we have, by Lemma

6.3.2, that c_s is scope-$[\bar{I}_d + s(\bar{n} - \bar{I}_d)]$. These presume that σ is maximal with respect to the configurations during synthesis. We must provide configurations at each step which are scope-$[\bar{I}_d + s(\bar{n} - \bar{I}_d)]$ having unique c(N(X,i)) in the configuration and on its boundaries. Then, maximality is insured (due to the size) and σ will exist (since all c(N(X,i)) are unique). Consider $(\bar{\ell},\bar{n})_d$ patterns embedded in a quiescent tessellation. Unfortunately, the c(N(X,i)) which border these patterns may not be unique. We will construct an $(\bar{\ell},\bar{n})_d$ pattern using a composition of two patterns. Initially, we construct two patterns, $p_1 = (\bar{\ell} - 2\bar{I}_d, \bar{n})_d$ and $p_2 = (MAX\bar{\ell} - 1, MIN\bar{n} - 1)_{d-1}$ using disjoint alphabets A_1 and A_2, respectively. Then p_1 is an irregular pattern and p_2 is a hypercube. The cardinality for A_1 is given by Theorem 5.4.7 and Theorem 5.3.5 gives a similar result for A_2. From p_2, we take d subpatterns, $p_{2,1}$, $p_{2,2}$, ..., $p_{2,i}$, ..., $p_{2,d}$, where $p_{2,1}$ is a $((\ell_2-1, \ell_3-2, \ldots, \ell_d-2), MIN\bar{n} - 1)_{d-1}$ pattern, $p_{2,2}$ is a $((\ell_1-2, \ell_3-1, \ell_4-2, \ldots, \ell_d-2, MIN\bar{n}-1)_{d-1}$ pattern, $p_{2,x}$ is a $((\ell_1-2,\ell_2-2, \ldots, \ell_{x-1}-2, \ell_{x+1}-1, \ell_{x+2}-2, \ldots, \ell_d-2), MIN\bar{n} - 1)_{d-1}$ pattern and $p_{2,d}$ is a $((\ell_1-1, \ell_2-2, \ldots, \ell_{d-1}-2), MIN\bar{n} - 1)_{d-1}$ pattern. More simply, we take a subpattern from p_2 which will concatenate with p_1 in dimension x, $1 \leq x \leq d$, but which is one unit larger than p_1 in dimension $x + 1$ where $x + 1$ is taken modulo-d. We concatenate $p_{2x} * p_1 * p_{2x}$ in dimension x such that the resultant pattern is span-$(\ell_1-2, \ldots, \ell_{x-1}-2, \ell_x, \ell_{x+1}, \ell_{x+2}-2, \ldots, \ell_d-2)$ and has Iscope-$(\ell_1-2, \ldots, \ell_{x-1}-2, \ell_x, \ell_{x+1}-2, \ldots, \ell_d-2)$. Clearly, if such a concatenation is made for all x, $1 \leq x \leq d$, the resultant will be span-$(\ell_1, \ell_2, \ldots, \ell_d)$ and have Iscope-$(\ell_1, \ell_2, \ldots, \ell_d)$. Then it is simply scope-$\bar{\ell}$.

Clearly, it is an $(\bar{\ell},\bar{n})_d$ pattern since p_1 is $(\bar{\ell} - 2\bar{I}_d, \bar{n})_d$, p_2 is $(MAX\bar{\ell} - 1, MIN\bar{n} - 1)_{d-1}$ and $A_1 \cap A_2 = \Phi$. (Φ is the empty set.) We consider the boundaries when the pattern is embedded in a quiescent tessellation. Since p_2 has all $MIN\bar{n} - 1$ subpatterns unique, $0 \notin A_1$, $0 \notin A_2$, then all $c(N(X,i))$ are unique on boundaries including those which overlap p_{2i}, p_{2j} and those containing 0. To construct p_1, p_2, we require

$$|A_2| \geq \left\lceil MIN\bar{n}-1 \sqrt{(MAX\bar{\ell}-1) - (MIN\bar{n}-1) + 1} \right\rceil,$$

$$|A_1| \geq \left\lceil MIN\bar{n} \sqrt{MAX(\bar{\ell}-2\bar{I}_d) - MIN\bar{n}+1} \right\rceil \text{ by Theorem 5.3.5 and}$$

Theorem 5.4.7, respectively. $|A_2|$ and $|A_1|$ may be rewritten as:

$$|A_2| \geq \left\lceil MIN\bar{n}-1 \sqrt{MAX\bar{\ell} - MIN\bar{n} + 1} \right\rceil,$$

$$|A_1| \geq \left\lceil MIN\bar{n} \sqrt{MAX\bar{\ell} - MIN\bar{n} - 1} \right\rceil.$$

Then, $|A_1| + |A_2|$ is sufficient to write an $(\bar{\ell},\bar{n})_d$ pattern which, when embedded in a quiescent tessellation has all $c(N(X,i))$ unique. At the s^{th} step, c_s is scope-$[\bar{I}_d + s(\bar{n}-\bar{I}_d)]$. Then,

$$\left\lceil MIN\bar{n}-1 \sqrt{MAX[\bar{I}_d+s(\bar{n}-\bar{I}_d) - MIN\bar{n} + 1} \right\rceil + \left\lceil MIN\bar{n} \sqrt{MAX[\bar{I}_d+s(\bar{n}-\bar{I}_d)] - MIN\bar{n} - 1} \right\rceil$$

is sufficient to write an $(\bar{\ell},\bar{n})_d$ pattern at the s^{th} step. This reduces to:

$$\left\lceil MIN\bar{n}-1 \sqrt{sMAX\bar{n} - MIN\bar{n} - s + 2} \right\rceil + \left\lceil MIN\bar{n} \sqrt{sMAX\bar{n} - MIN\bar{n} - s} \right\rceil.$$

To find $|A_s|$, we require two symbols for $c_o = c_p$, and simply sum from $s = 1$ to $t - 1$, $t = MAX\left\lceil \frac{\ell_i - 1}{n_i - 1} \right\rceil$. Then

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} \left\lceil \overline{MIN\overline{n}-1}\sqrt{sMAX\overline{n} - MIN\overline{n} - s+2} \right\rceil + \left\lceil \overline{MIN\overline{n}}\sqrt{sMAX\overline{n} - MIN\overline{n} - s} \right\rceil$$

where $t = MAX\left\lceil \frac{\overline{\ell_i - 1}}{n_i - 1} \right\rceil$, are sufficient to insure existence of σ.

The following example demonstrates the theorem.

Example 7.3.1 The span-(8,7) configuration below is synthesized in TA = (A, E^2, ((0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3), (2,0), (2,1), (2,2), (2,3)), σ) from a primitive configuration.

$$\begin{array}{c} \vdots \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0 \\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0 \\ \cdots\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ \cdots \\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0 \\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \vdots \end{array}$$

X is seen to be scope-(4,3). Then, by Lemma 6.3.2, c_s will be scope-(3s + 1, 2s + 1) and, by Corollary 6.3.4, c_f can be reached at t = $MAX\{\left\lceil \frac{\overline{8 - 1}}{4 - 1} \right\rceil, \left\lceil \frac{\overline{7 - 1}}{3 - 1} \right\rceil\} = 3$. We require only c_o which is primitive, c_1 which is scope-(4,3) and c_2 which is scope-(7,5). We select c_o as:

$$\begin{array}{ccccc} & & \vdots & & \\ & & 0\ \ 0\ \ 0 & & \\ \cdots & 0 & A & 0 & \cdots \\ & & 0\ \ 0\ \ 0 & & \\ & & \vdots & & \end{array}$$

and proceed to find c_1. To construct c_1, we require $p_1 = ((4,3) - (2,2), (4,3))_2 = ((2,1), (4,3))_2$ and $p_2 = (4\text{-}1, 3\text{-}1)_1 = (3,2)_1$. P_1 is absurd since its subpatterns are larger than itself but is trivially satisfied by any 2 by 1 pattern. Let p_1 = BC. For $p_2 = (3,2)_1$ we select DDE. Next, $p_{2,1}$ and $p_{2,2}$ are selected from p_2 where $p_{2,1} = (3\text{-}1, 3\text{-}1)_1 = (2, 2)_1$ and $p_{2,2} = (4\text{-}1, 3\text{-}1)_1 = (3,2)_1$. Let $p_{2,1}$ = DD and $p_{2,2}$ = DDE. These are concatenated with p_1 as described to yield:

$$\begin{array}{|c|ccc|} \hline D & D & D & E \\ \hline D & B & C & \multicolumn{1}{|c|}{D} \\ \cline{1-3} D & D & E & \multicolumn{1}{|c|}{D} \\ \cline{4-4} \end{array}$$

which is scope-(4,3) and will have all 4 by 3 subpatterns unique when embedded in a quiescent tessellation. Then, c_1 is to be:

$$\begin{array}{ccccccc} & & & \vdots & & & \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & D & D & D & E & 0 \\ \cdots & 0 & D & B & C & D & 0 & \cdots \\ & 0 & D & D & E & D & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & \vdots & & & \end{array}$$

It should be noted that this pattern does not employ the minimal number of symbols possible. For instance,

$$
\begin{array}{cccccccc}
 & & & \vdots & & & & \\
 & 0 & 0 & 0 & 0 & 0 & 0 & \\
 & 0 & B & B & B & B & 0 & \\
\cdots & 0 & B & B & B & B & 0 & \cdots \\
 & 0 & B & B & B & B & 0 & \\
 & 0 & 0 & 0 & 0 & 0 & 0 & \\
 & & & \vdots & & & &
\end{array}
$$

could be used. The construction was not claimed to use the minimal number of symbols and, as is seen, it does not.

Next, c_2 is found. Since it is to be scope-(7,5), we require $p_1 = ((7,5) - (2,2), (4,3))_2 = ((5,3), (4,3))_2$ and $p_2 = (7\text{-}1, 3\text{-}1)_1 = (6,2)_1$. P_1 is found by taking any 5 by 3 subpattern of a $(5,3)_2$ pattern. Using the techniques of Chapter V, we can find:

F F F G F

F F G F G

G F G G G

G G F F F

F G F G G

and select:

F F F G F

F F G F G

G F G G G

for p_1. For p_2, the string HHIHJH is utilized. Then, $p_{2,1} = (7\text{-}1, 3\text{-}1)_1$

$= (6,2)_1$ and $p_{2,2} = (5\text{-}1,\ 3\text{-}1)_1 = (4,2)_1$ are selected as HHIHJH and HHIH, respectively. These are concatenated with p_1 to find:

H	H	H	I	H	J	H
H	F	F	F	G	F	H
I	F	F	G	F	G	H
H	G	F	G	G	G	I
H	H	I	H	J	H	H

which is a $((7,5),\ (4,3))_2$ pattern and, when embedded in a quiescent tessellation:

				⋮					
	0	0	0	0	0	0	0	0	0
	0	H	H	H	I	H	J	H	0
	0	H	F	F	F	G	F	H	0
•••	0	I	F	F	G	F	G	H	0 •••
	0	H	G	F	G	G	G	I	0
	0	H	H	I	H	J	H	H	0
	0	0	0	0	0	0	0	0	0
				⋮					

has all c(N(X,i)) unique including boundaries. Due to the size of the neighborhood, σ will not be demonstrated here, but it should be clear that it does exist such that c_1, c_2, c_f is the actual sequence of global configurations in the automaton.

Corollary 7.3.2 In E^d TA, if X is scope-$\overline{n}$, $\xi_o \in X$, c_f is span-$\overline{\ell}$, c_o is scope-$\overline{r}$, $\forall i$, $n_i < r_i \leq \ell_i$,

$$|A_s| \geq 1 + \sum_{s=0}^{t-1} \left\lceil \overline{MIN}\overline{n} \sqrt{\overline{MAX}[\overline{r} + s(\overline{n}-\overline{I}_d)] - \overline{MIN}\overline{n} - 1} \right\rceil +$$

$$\left\lceil \overline{MIN}\overline{n}-1 \sqrt{\overline{MAX}[\overline{r} + s(\overline{n}-\overline{I}_d)] - \overline{MIN}\overline{n} + 1} \right\rceil,$$

where $t = \overline{MAX}\left\lceil \frac{\ell_i - r_i}{n_i - 1} \right\rceil$, then $\exists\, \sigma\ ^3\ c_f$ is synthesized from c_o.

Proof: By Theorem 6.3.3, we reach c_f at $t = \overline{MAX}\left\lceil \frac{\ell_i - r_i}{n_i - 1} \right\rceil$ and by Lemma 6.3.2, c_s is scope-$[\overline{r} + s(\overline{n}-\overline{I}_d)]$. By the same argument given in Theorem 7.3.1, we require

$$\left\lceil \overline{MIN}\overline{n} \sqrt{\overline{MAX}\ell - \overline{MIN}\overline{n} - 1} \right\rceil + \left\lceil \overline{MIN}\overline{n}-1 \sqrt{\overline{MAX}\ell - \overline{MIN}\overline{n} + 1} \right\rceil$$

symbols to construct an $(\overline{\ell},\overline{n})_d$ pattern having all c(N(X,i)) unique when embedded in a quiescent tessellation. Then, at the s^{th} step, we require:

$$\left\lceil \overline{MIN}\overline{n} \sqrt{\overline{MAX}[\overline{r} + s(\overline{n}-\overline{I}_d)] - \overline{MIN}\overline{n} - 1} \right\rceil + \left\lceil \overline{MIN}\overline{n}-1 \sqrt{\overline{MAX}[\overline{r} + s(\overline{n}-\overline{I}_d) - \overline{MIN}\overline{n} + 1} \right\rceil$$

symbols. One symbol is required for the quiescent symbol and we sum the required symbols for each step.

$$|A_s| \geq 1 + \sum_{s=0}^{t-1} \left\lceil \overline{MIN}\overline{n} \sqrt{\overline{MAX}[\overline{r} + s(\overline{n}-\overline{I}_d)] - \overline{MIN}\overline{n} - 1} \right\rceil +$$

$$\left\lceil \overline{MIN}\overline{n}-1 \sqrt{\overline{MAX}[\overline{r} + s(\overline{n}-\overline{I}_d)] - \overline{MIN}\overline{n} + 1} \right\rceil$$

insures the existence of σ where $t = \overline{MAX}\left\lceil \frac{\ell_i - r_i}{n_i - 1} \right\rceil$.

Corollary 7.3.3 In E^d TA, if X is scope-$\overline{n}$, $\xi_o \in X$, c_f is span-$\overline{\ell}$, c_o is scope-$\overline{r}$, $\forall i$, $r_i \leq n_i$,

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} \left\lceil \overline{MIN}\overline{n} \sqrt{s\overline{MAX}n - \overline{MIN}\overline{n} - s} \right\rceil + \left\lceil \overline{MIN}\overline{n}-1 \sqrt{s\overline{MAX}n - \overline{MIN}\overline{n} - s + 2} \right\rceil,$$

$t = MAX\left\lceil \frac{\ell_i - r_i}{n_i - 1} \right\rceil$, then $\exists\ \sigma \ni c_f$ is synthesized from c_o.

Proof: If $\forall i$, $r_i \leq n_i$, then only two symbols are required for c_o. We sum the remainder to find $|A_s|$.

As in the case of strings and one-dimensional automata, our results hold for neighborhoods which are span-$\bar{m}$, Iscope-$\bar{n}$. The following corollaries are proven trivially by considering only the Iscope-$\bar{n}$ portion of X.

Corollary 7.3.4 In E^d TA, if X is Iscope-$\bar{n}$, $\xi_o \in X$, c_f is span-$\bar{\ell}$, c_o is scope-$\bar{r}$, $\forall i$, $1 \leq r_i \leq n_i$,

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} \left\lceil \overline{MIN\bar{n}} \sqrt{sMAX\bar{n} - MIN\bar{n} - s} \right\rceil + \left\lceil \overline{MIN\bar{n}-1} \sqrt{sMAX\bar{n} - MIN\bar{n} - s + 2} \right\rceil,$$

where $t = MAX\left\lceil \frac{\ell_i - r_i}{n_i - 1} \right\rceil$, then $\exists\ \sigma \ni c_f$ is synthesized from c_o.

Corollary 7.3.5 In E^d TA, if X is Iscope-$\bar{n}$, $\xi_o \in X$, c_f is span-$\bar{\ell}$, c_o is scope-$\bar{r}$, $\forall i$, $n_i < r_i \leq \ell_i$,

$$|A_s| \geq 1 + \sum_{s=0}^{t-1} \left\lceil \overline{MIN\bar{n}} \sqrt{MAX\bar{r}+sMAX\bar{n}-MIN\bar{n}-s-1} \right\rceil + \left\lceil \overline{MIN\bar{n}-1} \sqrt{MAX\bar{r}+sMAX\bar{n}-MIN\bar{n}-s+1} \right\rceil,$$

$t = MAX\left\lceil \frac{\ell_i - r_i}{n_i - 1} \right\rceil$, then $\exists\ \sigma \ni c_f$ is synthesized from c_o.

Some effort has been expended to obtain a simple closed form result for the results of this section similar to that of Corollary 7.2.7 but the simultaneous presence of MAX and MIN in the expression for $|A_s|$ prevent us from finding a reasonable approximation. If X is restricted to being a hyper-cube, a closed form result can be found.

Corollary 7.3.5 For E^d TA, if X is scope-$\bar{n}$, $MAX\bar{n} = MIN\bar{n} = n > 1$, $\xi_o \in X$, c_o is scope-$\bar{r}$, c_f is span-$\bar{\ell}$, $|A_s| \geq t^2-t+2$, $t = \left\lceil \frac{MAX\bar{\ell} - 1}{n - 1} \right\rceil$, then $\exists\ \sigma \ni c_f$ is synthesized from c_o.

Proof: The result of Theorem 7.3.1 represents a worse case for $|A_s|$ since a non-primitive initial configuration will require fewer symbols to synthesize the same c_f. If $MAX\bar{n} = MIN\bar{n} = n$, this result reduces to:

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} \left\lceil \sqrt[n]{sn - n - s + 2} \right\rceil + \left\lceil \sqrt[n-1]{sn - n - s} \right\rceil.$$

By arguments similar to those preceding Corollary 7.2.4, we claim:

$$s^n \geq (sn - n - s + 2),\ s^{n-1} \geq (sn - n - s)\ \forall n \geq 1$$

where $s \geq 1$. Then,

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} (s) + (s)$$

$$|A_s| \geq 2 + \sum_{s=1}^{t-1} 2s$$

$$|A_s| \geq 2 + 2\left(\sum_{s=1}^{t-1} s\right)$$

$$|A_s| \geq 2 + (t-1)(t)$$

$$|A_s| \geq t^2 - t + 2$$

where $t = \left\lceil \frac{MAX\bar{\ell} - 1}{n - 1} \right\rceil$.

This result is easily extended to include arbitrary neighborhoods.

Theorem 7.3.6 For E^d TA, if X is Iscope-$\bar{n}$, $MIN\bar{n} > 1$, c_o is scope-$\bar{r}$, c_f is span-$\bar{\ell}$, $|A_s| \geq t^2-t+2$, $t = \left\lceil \frac{MAX\bar{\ell} - 1}{MIN\bar{n} - 1} \right\rceil$, then $\exists\ \sigma \ni c_f$ is synthesized from c_o.

Proof: If X is Iscope-$\bar{n}$, then it must contain a hyper-cube of size MIN$\bar{n}$ in each dimension. Then the results of Corollary 7.3.5 hold.

This concludes our formal investigation of synthesis in multi-dimensional tessellation automata. Theorem 7.3.6 is our most useful result since it provides an easily computed bound for $|A_s|$ which applies to automata of arbitrary dimension with nearly any neighborhood index. The cases not covered by it have Iscope-$\bar{n}$, MIN$\bar{n}$ = 1 which exhibit unpredictable growth. Of equal importance is Theorem 7.3.1 which defines the construction employing $(\bar{\ell},\bar{n})_d$ patterns for multi-dimensional synthesis.

7.4 Discussion

We have presented results which demonstrate the ability of tessellation automata to synthesize an arbitrary configuration if the alphabet utilized during synthesis meets or exceeds a lower bound. Suppose that a set of configurations, $S = \{c_1, c_2, \ldots\}$, $|S| = m$, are to be permitted and the automaton must be capable of synthesizing any of these. There are at least two approaches to this problem.

Perhaps the most straightforward of these is the following. For each c_i, $c_i \in S$, let there be a unique alphabet, A_{si}, and let $A_s = \bigcup_{i=1}^{m} A_{si}$. Then, there is a unique initial configuration, c_{oi}, and a partial definition of σ under which c_i is synthesized from c_{oi}. Clearly, there is no difficulty in defining σ since $\forall i,j$, $A_{si} \cap A_{sj} = \Phi$, $i \neq j$. Unfortunately, this approach causes A_s to grow by "leaps and bounds" as more configurations are added to S.

As an alternative to this difficulty, the following approach is considered. Each c_i, $c_i \in S$, has an associated d-tuple, $\bar{\ell}(i)$, where c_i

is span-$\bar{\ell}(i)$. Let L be a d-tuple, $L = (L_1, L_2, \ldots, L_d)$ where $L_j = MAX\{\ell_j(1), \ell_j(2), \ldots, \ell_j(m)\}$. We are merely defining a configuration, G, which is span-L where the span of G in dimension j is equal to the largest span (in dimension j) of any configuration found in S. We can construct σ which reaches a span-L configuration at some time t. This can be accomplished using the methods of our previous section. If this is done, then the configuration found in the automaton at time t-1 has all c(N(X,i)) unique. Since this is the case, σ may be chosen such that a given $c_i \in S$ arises at time t and since $\xi_o \in X$, we are assured that it can be an equilibrium configuration. In fact, if the automaton is made to be non-erasing for A_f, then any configuration composed of symbols in A_f will be an equilibrium configuration. Since $A_s \cap A_f = \Phi$, then there is no difficulty in defining σ so that this is the case. Then, if a different configuration in S is desired, only the portion of σ which moves TA from c_{t-1} to c_t need be redefined. (This was pointed out in the example for one-dimensional synthesis.) Then, the same sequence of configurations are employed to reach any configuration in S. We are, in a sense, relaxing our stipulation that σ be time-invariant. It will still be time-invariant for a given configuration in S which is to be synthesized but must be selected prior to initialization of the automaton.

Of the two methods considered, each has its advantage. If a unique alphabet is employed for synthesis of each configuration in S, only a single local transformation is required but a large alphabet is necessary. If a common synthesis sequence is employed, a relatively small alphabet is required but each cell must possess a set of local transformations.

Furthermore, an input which is common to each cell must be provided which selects the desired local transformation. If this input uses q-ary encoding, then $\lceil \log_q |S| \rceil$ signals are required. These can originate from a single register which is external to the tessellation automaton. Then, by placing the correct "word" in the register and "resetting" the automaton to the initial configuration will permit synthesis of any configuration in S.

VIII. PERTURBATION

8.1 Introduction

Within this chapter, the fault tolerant aspects of tessellation automata are considered. First, perturbation is defined.

Definition 8.1.1 A next state configuration, c', is said to be perturbed if and only if $\exists$ a cell, i, $c'(i) \in A$ but $c'(i) \neq \sigma(c(N(X,i)))$.

Informally, a configuration is "perturbed" if some cell made an incorrect transition at the previous time step. It is required that the cell fail to a valid symbol in the automaton's alphabet. Furthermore, some restriction must be made as to the number of simultaneous perturbations permitted. For the most part, only single perturbations will be considered. Most results will hold if the perturbations are separated sufficiently in the automaton.

Having defined perturbation, we consider what, if anything, can be done should one occur. There are basically two areas of concern. First, the automaton must detect the fact that it has moved to an incorrect configuration and, hopefully, can ascertain which cell or cells are perturbed. This will be referred to as self-diagnosis. Secondly, the automaton should move to a configuration which is "equivalent" to the one which should have occurred but does not utilize the cells which were perturbed. This will be referred to as self-reconfiguration. Clearly, this implies a great deal, and we must question the solvability of such a proposal since it closely resembles the stability question. Fortunately, we do not address arbitrary automata. Furthermore, a perturbed con-

figuration is not completely unknown if the number of simultaneous perturbations is limited. Clearly, the questions of self-diagnosis and self-reconfiguration differ somewhat from the stability question. For that reason, it may be solvable. Our argument does not demonstrate solvability but it does not rule out the possibility either.

8.2 Restoration of Equilibrium Configurations

In this section, the possibility of restoring perturbed cells to their correct configuration is considered. The following definitions formalize this notion.

Definition 8.2.1 A configuration, c, in a tessellation automaton, TA, is said to be an immediately restorable configuration (IRC) for TA if and only if for any perturbation of c, c*, $\tau(c^*) = c$, and $\tau(c) = c$.

Definition 8.2.2 A configuration, c, in a tessellation automaton, TA, is said to be a sequentially restorable configuration (SRC) for TA if and only if for any perturbation of c, c*, $\tau^k(c^*) = c$, for some k and $\tau(c) = c$.

Sequentially restorable configurations will not be investigated and the definition is offered only to acknowledge their possible existence. The following example demonstrates the difficult nature of immediate restoration.

Example 8.2.1 Consider an E^1 TA with $X = (-1,0,1)$ and $A = \{0,1,2,3,4\}$. Suppose that TA is in a final configuration, $c_f = \overline{0}1234440\overline{0}$. Then σ is partially specified.

σ: 0 0 0 - 0

0 0 1 - 0

0 1 2 - 1

1 2 3 - 2

2 3 4 - 3

3 4 4 - 4

4 4 4 - 4

4 4 0 - 4

4 0 0 - 0

The problem is to define (if possible) the remaining elements of σ in such a way that if any single perturbation occurs, the array will return to c_f at the next time step. If, for example, the cell in state "2" is perturbed to state "4", it must return to "2" at the next time step. Then,

$$c_f = \overline{0}\ 1\ 2\ 3\ 4\ 4\ 4\ \overline{0}$$
$$c_f^* = \overline{0}\ 1\ 4\ 3\ 4\ 4\ 4\ \overline{0}$$
$$c' = \overline{0}\ 1\ 2\ 3\ 4\ 4\ 4\ \overline{0}$$

This feat requires that $\sigma(143) = 2$ in order that the cell's correct state be restored. Notice that the cells in state "1" and "3" have undefined neighborhoods in c_f^*. If $\sigma(014) = 1$ and $\sigma(434) = 3$, then the neighbors of the perturbed cell retain their correct state. This is clearly a necessary condition if c is to be IRC for TA. As σ is currently defined, the configuration will be restored for the particular perturbation considered. Unfortunately, repeated application of this technique will not work. Consider the requirements of σ if the second cell in state "4" is

perturbed to state "3".

$$c_f = \overline{0}\ 1\ 2\ 3\ 4\ 4\ 4\ \overline{0}$$
$$c_f^* = \overline{0}\ 1\ 2\ 3\ 4\ 3\ 4\ \overline{0}$$
$$c_f = \overline{0}\ 1\ 2\ 3\ 4\ 4\ 4\ \overline{0}$$

To achieve c_f following c_f^*, it is required that $\sigma(234) = 3$, $\sigma(434) = 4$, $\sigma(340) = 4$. Recall that $\sigma(434) = 3$ has been previously defined. It can be concluded that c_f cannot be an IRC for TA.

For the example, it may be possible that a larger neighborhood or a modification of the configuration will lead to immediate restoration.

Definition 8.2.2 A configuration, c, which undergoes a single perturbation where the j^{th} cell is perturbed to $* \in A$ will be referred to as c^{j*}.

Following this notation, the neighborhood of the perturbed cell, j, is $c^{j*}(N(X,j))$, and the neighborhood of the i^{th} cell is $c^{j*}(N(X,i))$. Clearly, if c is to be an IRC, it is both necessary and sufficient that $\forall i,j, *, \sigma(c^{j*}(N(X,i))) = c(i)$. Unfortunately, if $\exists\ i_1, j_1, *_1, i_2, j_2, *_2 \ni c^{j_1*_1}(N(X,i_1)) = c^{j_2*_2}(N(X,i_2))$ but $c(i_1) \neq c(i_2)$, then c cannot be an IRC for TA. The following theorem formalizes this discussion.

Theorem 8.2.1 For a given configuration, c, $\exists\ \sigma \ni$ c is an IRC if and only if $\nexists\ i_1, j_1, *_1, i_2, j_2, *_2 \ni c^{j_1*_1}(N(X,i_1)) = c^{j_2*_2}(N(X,i_2))$ and $c(i_1) \neq c(i_2)$.

Proof: First, σ is constructed. $\forall i, j, *$, let $\sigma(c^{j*}(N(X,i))) = c(i)$. If $\exists\ i_1, j_1, *_1, i_2, j_2, *_2, \ni c^{j_1*_1}(N(X,i_1)) = c^{j_2*_2}(N(X,i_2))$, then

$c(i_1) = c(i_2)$. Therefore, σ exists. Restoration of c under σ is demonstrated next. Let cell i be perturbed to *. $\exists$ n cells at locations $i_1 = i - \xi_1$, $i_2 = i - \xi_2$, ..., $i_n = i - \xi_n$, which have perturbed neighborhood configurations. For these cells,

$$c^{i*}(N(X,i_1)) = * \quad c(i_1 + \xi_2) \quad c(i_1 + \xi_3) \ldots c(i_1 + \xi_n)$$

$$c^{i*}(N(X,i_2)) = c(i_2 + \xi_1) \quad * \quad c(i_2 + \xi_3) \ldots c(i_2 + \xi_n)$$

$$\vdots$$

$$c^{i*}(N(X,i_n)) = c(i_n + \xi_1) \quad c(i_n + \xi_2) \ldots c(i_n + \xi_{n-1}) \quad *$$

Since $\forall i, j, *, \sigma(c^{j*}(N(X,i))) = c(i)$, then $\sigma(c^{i*}(N(X,i_1))) = c(i_1)$, $\sigma(c^{i*}(N(X,i_2))) = c(i_2)$, ..., $\sigma(c^{i*}(N(X,i_n))) = c(i_n)$. Then, $\tau(c^{i*}) = c$. This concludes sufficiency. If $\exists\ i_1, j_1, *_1, i_2, j_2, *_2, \ni c^{j_1*_1}(N(X,i_1)) = c^{j_2*_2}(N(X,i_2))$, $c(i_1) \neq c(i_2)$, then it is necessary that $\sigma(c^{j_1*_1}(N(X,i_1))) = c(i_1)$, $(c^{j_2*_2}(N(X,i_2))) = c(i_2)$. Then σ is multivalued and cannot exist. This concludes necessity.

The conditions of the theorem can be separated to produce two sufficient but not necessary requirements.

Corollary 8.2.2 If $\nexists\ i_1, j_1, *_1, i_2, j_2, *_2, \ni c^{j_1*_1}(N(X,i_1)) = c^{j_2*_2}(N(X,i_2))$, then $\exists\ \sigma \ni c$ is an IRC.

Corollary 8.2.3 If $\forall i, j$, $c(i) = c(j)$, then $\exists\ \sigma \ni c$ is an IRC.

The following result follows directly from the second corollary.

Theorem 8.2.4 $\forall X$, $\exists\ \sigma \ni c = \bar{a}$ is an IRC.

The following results consider successively larger neighborhoods.

Corollary 8.2.5 If $|X| = 1$, then $c = \bar{a}$ is the only IRC.

Proof: By Theorem 8.2.4, $c = \bar{a}$ is an IRC, $\forall X$. Then, it is an IRC if $|X| = 1$. Only uniqueness must be demonstrated. Let c be a configuration, $c \neq \bar{a}$. Then $\exists$ i $\ni$ $c(i) \neq c(i+1)$. Since $|X| = 1$, then $X = (\xi)$. It is required that $\sigma(c(N(X,i))) = c(i)$, $\sigma(c(N(X,i+1))) = c(i+1)$ so that c be in equilibrium. Since $|X| = 1$, then $c(N(X,i)) = c(i+\xi)$, $c(N(X,i+1)) = c(i + 1 + \xi)$. Then, $\sigma[c(i+\xi)] = c(i)$, $\sigma[c(i + 1 + \xi)] = c(i + 1)$. If $c(i + \xi) = c(i + 1 + \xi)$, then σ will not exist since $c(i) \neq c(i + 1)$. Therefore, it is required that $c(i + \xi) \neq c(i + 1)$ if c is to be in equilibrium. Consider the perturbation of c where the cell at location $i + \xi$ is perturbed to $c(i + \xi + 1)$. If the cell at location i is to retain its content, then it is required that $\sigma[c(i + \xi + 1)] = c(i)$. But $\sigma[c(i + \xi + 1)] = c(i + 1)$, and $c(i + 1) \neq c(i)$. Then $\nexists$ σ. Therefore, $c = \bar{a}$ is the only IRC if $|X| = 1$.

Corollary 8.2.6 If $|X| = 2$, then $c = \bar{a}$ is the only IRC.

Proof: That $c = \bar{a}$ is an IRC if $|X| = 2$ has been demonstrated. Only uniqueness must be proven. Let c be a configuration, $c \neq \bar{a}$. Then, $\exists$ i $\ni$ $c(i) \neq c(i + 1)$. Since $|X| = 2$, $X = (\xi_1, \xi_2)$. For equilibrium of c, it is required that $\sigma(c(N(X,i))) = c(i)$ and $\sigma(c(N(X,i+1))) = c(i + 1)$. Since $c(N(X,i)) = c(i + \xi_1)c(i + \xi_2)$ and $c(N(X,i+1)) = c(i + 1 + \xi_1)$ $c(i + 1 + \xi_2)$, then it is necessary that either $c(i + \xi_1) \neq c(i + 1 + \xi_1)$ or $c(i + \xi_2) \neq c(i + 1 + \xi_2)$. Otherwise, σ will not exist. There are three cases which satisfy this criteria.

Case I. $c(i + \xi_1) \neq c(i + 1 + \xi_1)$, $c(i + \xi_2) = c(i + 1 + \xi_2)$. Let the

cell at $i + \xi_1$ be perturbed to $c(i + 1 + \xi_1)$. Then $c^*(N(X,i)) = c^*(N(X, i + 1))$, and it is required that $\sigma(c^*(N(X,i))) = c(i)$ and $\sigma(c^*(N(X,i + 1))) = c(i + 1)$. But $c(i) \neq c(i + 1)$ so σ cannot exist.

Case II. $c(i + \xi_2) \neq c(i + 1 + \xi_2)$, $c(i + \xi_1) = c(i + 1 + \xi_1)$. By the same argument as Case I, σ cannot exist.

Case III. $c(i + \xi_1) \neq c(i + 1 + \xi_1)$, $c(i + \xi_2) \neq c(i + 1 + \xi_2)$. Two different perturbations must be considered. Let the cell at location $i + \xi_2$ be perturbed to $c(i + 1 + \xi_2)$. Then $c^*(N(X,i)) = c(i + \xi_1)$ $c(i + 1 + \xi_2)$ and $\sigma[c(i + \xi_1)c(i + 1 + \xi_2)] = c(i)$ is required. Next, let the cell at location $i + 1 + \xi_1$ be perturbed to $c(i + \xi_1)$. (This is a different perturbation of c, not a simultaneous perturbation of two cells in c.) Then $c^*(N(X,i + 1)) = c(i + \xi_1)c(i + 1 + \xi_2)$ and it is necessary that $\sigma[c(i + \xi_1)c(i + 1 + \xi_2)] = c(i + 1)$. Since $c(i) \neq c(i + 1)$, this implies that σ does not exist since $\sigma[c(i + \xi_1)c(i + 1 + \xi_2)] = c(i)$ is required to restore the previous perturbation. Then $c = \overline{a}$ is the unique IRC if $|X| = 2$.

Corollary 8.2.7 If $|X| \geq 3$, then $c = \ldots a_{i_j} a_{i_{j+1}} a_{i_{j+2}} \ldots$ where $\forall$ k, $k \neq j$, $a_{i_j} \neq a_{i_k}$, is an IRC.

Proof: Since all symbols are unqiue, then $\forall i, j$, $i \neq j$, $c(N(X,i)) \neq c(N(X,j))$. It is clear that for any perturbation, c^*, $\forall i, j$, $i \neq j$, $c^*(N(X,i)) \neq c^*(N(X,j))$. Then, by Theorem 8.2.1, c is an IRC.

This concludes findings for restoration. Although further results might be obtained, it is clear that a large number of symbols are required to retain unique $c^*(N(X,i))$ for any perturbation even with larger neighbor-

hoods. Furthermore, restoration is not generally useful for two reasons. If a cell is perturbed, it is probably faulty and cannot (or should not) be restored. Furthermore, the array is presumed to be at equilibrium for these results. This, too, is not a desirable assumption. If the array is not in equilibrium, restoration is much more complicated since the perturbed cell should not be restored to its previous correct state in c but should be placed in the correct state for c'. For these reasons, restoration was abandoned. It did serve some purpose, however, since the insight gained by this study greatly aided in the work which follows.

8.3 Self-Diagnosis of Single Perturbations

In this section, results relating to self-diagnosis in tessellation automata are presented. As stated previously, self-diagnosis implies that the cell or cells which have been perturbed are identified in some fashion. The following definitions formalize this notion.

Definition 8.3.1 A perturbed cell, i, in c is said to be quarantined in c' if and only if $\forall \xi \in X$, $\xi \neq \xi_o$, $c'(i - \xi) = Q$, where Q is a special symbol in A called the quarantine symbol.

Notice that if $\xi_o \in X$, any cell has n - 1 neighbors (other than itself). If a cell is perturbed, these n - 1 neighbors must go to the quarantine state, Q, at the next time step. If $\xi_o \notin X$, there are n neighbors which must reach Q to quarantine a perturbed cell.

Definition 8.3.2 A configuration, c, in some tessellation automaton, TA, is said to be one-step diagnosable by TA if and only if for any

perturbation of c, the perturbed cell(s) are quarantined in c'.

Definition 8.3.3 A tessellation automaton, TA, is said to be self-diagnosing on $D \subseteq C$ if and only if $\forall c \in D$, c is diagnosable by TA and $\tau(c) \in D$.

Note that self-diagnosing tessellation automata are not required to diagnose all possible configurations. They do diagnose all configurations that can arise if the automaton's initial configuration is in D. This follows from the fact that $\forall c \in D$, $\tau(c) \in D$. Then D is closed under τ and is a subautomaton of TA.

All of the results to be presented hold for single perturbations. For that reason, when an automaton is classed as self-diagnosing, it is implicit that this is true only for single perturbations.

The following definition permits the consideration of finite tessellation automata.

Definition 8.3.4 A finite d-dimensional tessellation space E_L^d is a finite subset of E^d specified by a d-tuple, $L = (\ell_1, \ell_2, \ldots, \ell_d)$. A d-tuple, $p = (p_1, p_2, \ldots, p_d)$ is in E_L^d if and only if $\forall i$, $0 \leq p_i \leq \ell_i - 1$.

A finite tessellation automaton, $TA = (A, E_L^d, X, \sigma)$ has cells which have undefined neighbors. That is, $\exists\, i, \xi \ni i + \xi \notin E_L^d$. It is assumed that these boundary inputs are, for all time, the quiescent symbol. The same assumption is made for $TA = (A, E_L^d, X, I)$.

Definition 8.3.5 A cyclic tessellation space, $S_L^d = (E_L^d, H)$ is specified by a finite tessellation space E_L^d and a d-tuple of d-tuples, $H = (h_1, h_2,$

..., h_d). The neighbor of a cell specified by $i + \xi$, $\xi = (z_1, z_2, \ldots, z_d)$, $\forall i$, $|z_i| < \ell_i - 1$, where $i + \xi \notin E_L^d$ is found using H as follows. Let $(i + \xi)_j$ be the j^{th} component of the d-tuple $(i + \xi)$. Let $R_{i,\xi} = \{\pm h_j \in H | (i + \xi)_j > \ell_j - 1 \Rightarrow + h_j \in R_{i,\xi}, (i + \xi)_j < 0 \Rightarrow - h_j \in R\}$ and let $m = \sum_{\forall h_j \in R_{i,\xi}} h_j$. Then the neighbor of cell i found by $i + \xi$ is taken to be $(i + \xi + m)$ where $\forall k$, the k^{th} component is taken modulo-ℓ_k.

It should be clear that for any $(k^n,n)_d$ pattern which is cyclic in all dimensions there is a cyclic tessellation space, S_L^d, which is cyclic in the same sense as the pattern. For these particular spaces, H has the property that h_x is the d-tuple of all zeroes and that all other elements of H are zero in all positions except the x^{th}. The x^{th} position may be zero as well but it is the only position which can be non-zero. (This property follows directly from consideration of the proof for Corollary 5.3.2 where it was shown that ℓ and x_1 differ only in the x^{th} component plus the fact that the pattern is cyclic in the usual sense in the x^{th} dimension.) To emphasize this point and to explain the definition, a small cyclic space is constructed for the pattern following Corollary 5.3.2. The pattern is shown below.

y \ x	0	1	2	3
0	0	0	1	1
1	0	1	1	0
2	1	0	0	1
3	1	1	0	0

This is a $(4,2)_2$ pattern which is cyclic in all dimensions if the adjacency is chosen appropriately. Since the pattern is of size four by four and has two dimensions, then d = 2, L = (4,4) are required. Then $S^2_{(4,4)}$

$= (E^2_{(4,4)}, H)$ where H must be specified to retain the correct cyclic adjacency. If H = ((0,0), (2,0)) is used, this requirement will be met. To demonstrate this point, let X = ((0,0), (0,1), (1,0), (1,1)) and compute the neighborhood configuration of the cell at (3,3). For ξ_o = (0,0), $i + \xi_o = (3,3)$ and no computation is required. For $\xi_1 = (0,1)$, $i + \xi_1 = (3,4)$. Since $\ell_2 = 4$, then $R_{i,\xi} = \{h_2\} = \{(2,0)\}$ and m = (2,0). Then $i + \xi_1 + m = (5,4)$ and, if each component is taken modulo-4, the neighbor using ξ_1 is at (1,0). For $\xi_2 = (1,0)$, $i + \xi_2 = (4,3)$ and $R = \{h_1\} = \{(0,0)\}$, m = (0,0). Then $i + \xi_2 + m = (4,3)$ and taken modulo-4, finds (0,3) as the neighbor of (3,3) using ξ_2. Finally, $\xi_3 = (1,1)$ is considered. For it, $i + \xi_3 = (4,4)$ so $R = \{h_1, h_2\} = \{(0,0), (2,0)\}$ and m = (2,0). Then (6,4) is taken modulo-4 in each component to find (2,0) as the neighbor for ξ_3. Then cell (3,3) has neighbors at (3,3), (1,0), (0,3), (2,0). If the adjacency found previously (following Corollary 5.3.2) is considered, these results are confirmed. Finally, the neighborhood configuration for cell (3,3) is assembled to find:

0 1
0 1

To further demonstrate the definition, suppose some neighborhood index has an element $\xi = (-2,-2)$. In this event, the neighbor of cell (0,0) is found by: $i + \xi = (-2,-2)$, $R_{i,\xi} = \{h_1, h_2\} = \{(0,0),-(2,0)\}$, $m = -(2,0)$, $i + \xi + m = (-4,-2)$ taken modulo-4 yields (0,2). This, too, can be verified by moving -2 in dimension x and -2 in dimension y (or vice versa). As a further demonstration of the definition, the final pattern found in the discussion following Theorem 5.3.1 is considered. The pattern found was:

```
                                        1 1 0 0
                                        1 0 0 1
                              1 0 0 1   0 1 1 0
                              0 0 1 1   0 0 1 1
                    0 1 1 0   1 1 0 0
                    1 1 0 0   0 1 1 0
          0 0 1 1   0 0 1 1
          0 1 1 0   1 0 0 1
          1 0 0 1
y         1 1 0 0
|_ z
 \
  x
```

This pattern is cyclic in all dimensions for the space $(E^3_{(4,4,4)}$, ((0,0,0), (2,0,0), (2,0,0))). To demonstrate, the neighbor of cell (3,3,3) is found for $\xi = (1,1,1)$. In this case, $i + \xi = (4,4,4)$, $R_{i,\xi} = \{h_1, h_2, h_3\} = \{(0,0,0), (2,0,0), (2,0,0)\}$, $m = (4,0,0)$, $i + \xi + m = (8,4,4)$ taken modulo-4 in each component to find (0,0,0). To further demonstrate, if $\binom{00}{01}$ is to be adjacent to itself between the first and last patterns, then the cell at (2,3,3) must be adjacent to (0,3,0) if $\xi = (0,0,1)$. To check this, $i + \xi = (2,3,4)$, $R_{i,\xi} = \{h_3\} = \{(2,0,0)\}$, $m = (2,0,0)$, $i + \xi + m = (4,3,4)$ taken modulo-4 to find (0,3,0) as expected.

Notice that the definition does not apply if the elements of ξ cause a boundary to be crossed more than once. It is possible to modify the definition to include these cases by counting the number of times a boundary will be crossed by $\lfloor (i + \xi)_j / (\ell_j - 1) \rfloor$ and including h_j in m that many times. An automaton with elements in X which "reach around" the array one or more times has a poorly chosen neighborhood. Further-

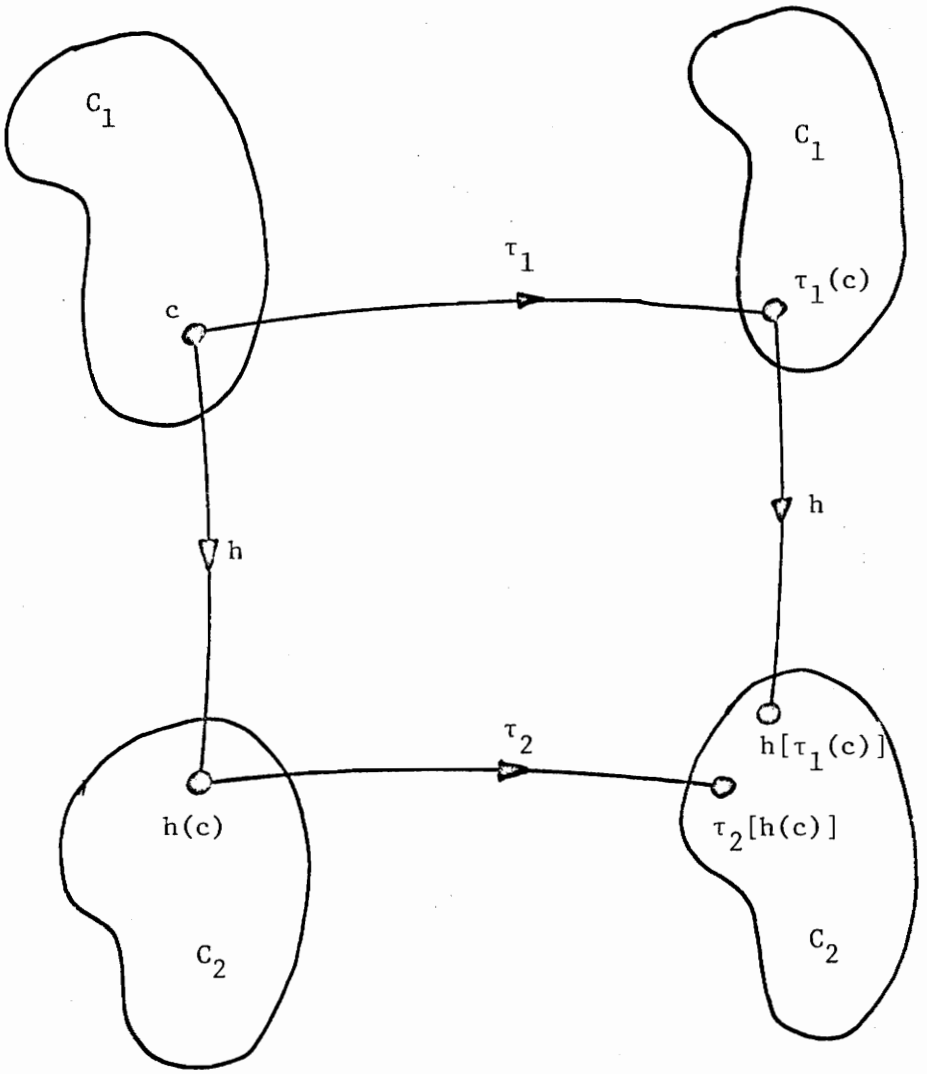
more, for any of these, there is an equivalent neighborhood which fits the definition. For instance, $\xi=(-7,10)$ is exactly equivalent to $\xi=(1,2)$ since they find the same neighbor. This definition permits discussion of a new class of tessellation automata, $TA=(A,S_L^d,X,I)$ and their less general counterparts, autonomous tessellation automata, $TA=(A,S_L^d,X,\sigma)$. Both classes have a finite number of cells since the space S_L^d is bounded. It is these automata on which the remainder of this chapter focuses.

Definition 8.3.6 Two tessellation automata, TA_1, TA_2, are said to be homomorphic if and only if $\exists\, h\colon C_1 \to C_2$ where $\forall c \in C_1\ h[\tau_1(c)] = \tau_2(h[c])$. Under this condition, TA_2 is said to be the homomorphic image of TA_1.

Clearly, two tessellation automata which are homomorphic do, in a strong sense, the same computation if $\forall c_i \in C_2$, $\exists\ c_j \in C_1 \ni h(c_j) = c_i$. This follows from the fact that all states (global) in C_2 are "covered" in C_1 and that homomorphic global transitions are made. An example of homomorphism between two TA is shown in Figure 8.3.1.

Lemma 8.3.1 $\forall TA = (A, E_L^d, X, \sigma)$ and $\forall k > 1$, $\exists$ a homomorphic TA, $TA_h = (A_h, S_{k2}^1, (-1,0,1), \sigma_h)$ which is self-diagnosing.

Proof: Since TA is in E_L^d, the set of configurations, C, is finite. Let $r = |C|$ and construct a set of configurations for TA_h, $D = \{d_1, d_2, \ldots, d_r\}$ where $\forall i$, d_i is a $(k^2,2)$ cyclic string on a set of symbols A_i, $|A_i| = k$. Furthermore, $\forall i, j$, $i \neq j$, $A_i \cap A_j = \Phi$. Then $|C| = |D|$ and the automaton TA_h must be in the space S_{k2}^1. Note that if $k < 1$, self-diagnosis is not possible since there are not enough cells to quarantine



TA_1 and TA_2 are homomorphic if and only if

$$\tau_2[h(c)] = h[\tau_1(c)], \forall c \in C_1.$$

Fig. 8.3.1 Two homomorphic tessellation automata

a perturbation. Let $A_h = \{Q\} \bigcup_{i=1}^{r} A_i$. Define an arbitrary bijective mapping h: $D \rightarrow C$. Then, $\forall d \in D, \exists c \in C \ni h(d) = c$ and $\forall c \in C, d \in D \ni h^{-1}(c) = d$. $\forall i$, let $\tau_h(d_i) = d_j$ if and only if $h(d_i) = c_i$, $h(d_j) = c_j$ and $\tau(c_i) = c_j$. Since $\forall d_i \in D$, d_i is a $(k^2,2)$ cyclic string using a unique alphabet, A_i, then $\exists \sigma_h \ni \tau_h(d_i) = d_j$. Since $\tau_h(d_i) = d_j$, then $h[\tau_h(d_i)] = h(d_j)$, but $h(d_j) = c_j$. Therefore, $h[\tau_h(d_i)] = c_j$. Since $h(d_i) = c_i$, and $\tau(c_i) = c_j$, then $\tau[h(d_i)] = c_j$. Therefore, $h[\tau_h(d_i)] = \tau[h(d_i)]$, $\forall i$. Then TA is the homomorphic image of TA_h. It should be noted that $D \subset C_h$ where C_h is the set of all configurations of TA_h. Furthermore, σ_h is not yet completely specified since $\exists p$, $p \in A_h^3$, where $\forall i, j$, $d_i(N(X,j)) \neq p$. That is, some elements of A_h^3 do not appear in any configuration in D. For these, let $\sigma(p) = Q$. Consider the perturbation of cell i in configuration d_j where $d_j(i) = *$, $* \in A_h$. There exist three cells at locations i - 1, i, i + 1 with perturbed neighborhood configurations. For these cells, $d_j(N(X,i-1)) = d_j(i-2)\ d_j(i-1)\ *$, $d_j(N(X,i))) = d_j(i-1) * d_j(i+1)$, $d_j(N(X,i+1)) = * d_j(i+1)\ d_j(i+2)$. Since d_j is a $(k^2,2)$ cyclic string using a unique alphabet, A_j, then $(d_j(i-2)d_j(i-1))$ and $(d_j(i+1)d_j(i+2))$ are unique substrings of length two. Then $d_j(N(X,i-1))$ and $d_j(N(X,i+1))$ are not in D since $* \neq d_j(i)$. Then, clearly, $\sigma(d_j(i-2)d_j(i-1)*) = Q$, $\sigma(* d_j(i+1)d_j(i+2)) = Q$ if $* \in A_h$. The next state of cell i is not known since it is presumed faulty but its neighbors at i-1, i+1, will go to Q at the next time step. For all other cells, $k \neq i-1, i, i+1$, the neighborhood configuration $d_j(N(X,k))$ is not perturbed. Therefore, $d_j'(k) \neq Q$. (These cells make their normal transition.) Then d_j is one-step diagnosable by TA_h. Since the preceding argument is for arbitrary j, then TA_h is self-diagnosing on D. It should

be clear that TA_h is isomorphic to TA if the restricted set of configurations D is considered. This follows from the fact that h is bijective between C and D. The lemma is demonstrated next.

Example 8.3.1 Consider the following TA, TA = ($\{0,1\}$, E_3^1, (0,1), σ) where:

$$\sigma(00) = 0$$

$$\sigma(01) = 1$$

$$\sigma(10) = 1$$

$$\sigma(11) = 0$$

For TA, there exist eight configurations.

i	c_i
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

τ may be computed from σ where:

i	c_i	$c_j=\tau(c_i)$	j
0	0 0 0	0 0 0	0
1	0 0 1	0 1 1	3
2	0 1 0	1 1 0	6
3	0 1 1	1 0 1	5
4	1 0 0	1 0 0	4
5	1 0 1	1 1 1	7
6	1 1 0	0 1 0	2
7	1 1 1	0 0 1	1

In calculating $\tau(c_i)$ we presumed that inputs required under X from non-existent cells are 0. Only cell three has this requirement. For TA_h, the space S_4^1 is employed. Then (4,2) cyclic strings can be used as configurations in D for TA_h. Selected are:

i	d_i
0	A A B B
1	C C D D
2	E E F F
3	G G H H
4	I I J J
5	K K L L
6	M M N N
7	P P R R

The mapping h is defined next.

i	d_i	$c_i=h(d_i)$
0	A A B B	0 0 0
1	C C D D	0 0 1
2	E E F F	0 1 0
3	G G H H	0 1 1
4	I I J J	1 0 0
5	K K L L	1 0 1
6	M M N N	1 1 0
7	P P R R	1 1 1

Next, τ_h is defined.

i	d_i	$d_j = {}_h(d_i)$	j
0	A A B B	A A B B	0
1	C C D D	G G H H	3
2	E E F F	M M N N	6
3	G G H H	K K L L	5
4	I I J J	I I J J	4
5	K K L L	P P R R	7
6	M M N N	E E F F	2
7	P P R R	C C D D	1

Finally, σ_h is found.

A A B - A	I I J - I
A B B - B	I J J - J
B B A - B	J J I - J
B A A - A	J I I - I
C C D - G	K K L - P
C D D - H	K L L - R
D D C - H	L L K - R
D C C - G	L K K - P
E E F - M	M M N - E
E F F - N	M N N - F
F F E - N	N N M - F
F E E - M	N M M - E
G G H - K	P P R - C
G H H - L	P R R - D
H H G - L	R R P - D
H G G - K	R P P - C

all others - Q

If σ_h is examined, it should be clear that TA_h will make the same transitions as TA. For instance:

TA: 010 → 110 → 010 → 110 → etc.

TA_h: EEFF → MMNN → EEFF → MMNN → etc.

To demonstrate the self-diagnosis of TA_h, consider the configuration EEFF where the third cell is perturbed to E. Then,

TA_h: EEEF → MQMQ

Then the third cell is diagnosed. (Actually, both cell one and cell three are diagnosed. This is due to the fact that the space is of length four in TA_h. A larger space would not present this difficulty.) Suppose cell two in MMNN were perturbed to A. Then:

TA_h: MANN → QQQF

It should be clear that any perturbation of any cell in any d_i yields a configuration not in D. In each of these perturbed configurations the two cells on either side of the perturbation have neighborhood configurations which lead them to Q at the next transition.

The lemma can be extended to TA with more than one local transformation.

Theorem 8.3.2 $\forall$ TA = (A, E_L^d, X, I) and $\forall k > 1$, $\exists$ a homomorphic TA, TA_h = (A_h, S_{k2}^1, (-1,0,1), I_h) which is self-diagnosing.

Proof: Recall that $I = \{\tau_i\}$ where τ_i is implied by σ_i. It is clear that the construction of Lemma 8.3.1 can be applied to each σ_i to yield the homomorphic TA_h which is self-diagnosing where $\forall \sigma_i$ in TA, $\exists$ σ_{ih} in TA_h.

Corollary 8.3.3 $\forall$ TA = (A, S^{d}_{L}, X, I) and $\forall k > 1$, $\exists$ a homomorphic TA, TA_h = (A_h, $S^{1}_{k}2$, (-1,0,1), I_h) which is self-diagnosing.

Proof: Since any TA in S^{d}_{L} has a finite number of configurations, then Theorem 8.3.2 holds.

The automaton TA = (A, E^{1}_{L}, X, I) permits simulation of finite state machines (FSM) where, for each input to the FSM, there is a corresponding σ selected by the input to TA. From this follows an important result. First, the notation to be used for finite-state machines is formalized.

Definition 8.3.6 A finite-state machine (FSM) is formalized by a five-tuple, (I, Q, Z, δ, ω) where I is a set of input symbols, Q is a set of states, Z is a set of output symbols, δ is the next state function δ: Q x I $\rightarrow$ Q and ω is the output function, ω: Q x I $\rightarrow$ Z.

If the output function, ω, depends on I, the FSM is said to be a Mealy machine. If ω is dependent only on Q, then the FSM is said to be a Moore machine. This notation for the FSM will be employed for the remainder of this dissertation.

Corollary 8.3.4 $\forall$ FSM, $k > 1$, $\exists$ a homomorphic TA = (A, $S^{1}_{k}2$, (-1,0,1), I) which is self-diagnosing.

Proof: Recall that for an FSM, δ: Q x I $\rightarrow$ Q, ω: Q x I $\rightarrow$ Z for Mealy machines, ω: Q $\rightarrow$ Z for Moore machines. Consider δ and recall the proof of Lemma 8.3.1. $\forall q \in Q$, select a different $d \in D$ and let h(d) = q. $\forall x \in$ I, define τ_x as follows: if $\delta(p,x) = q$ for some $p,q \in Q$, and if h(d) = p,

$h(e) = q$, then let $\tau_x(d) = e$. Then, an input x to the FSM corresponds to the global transformation τ_x. By a construction identical to that of Lemma 8.3.1, TA can be obtained that emulates the state structure of the FSM. Outputs from the TA that correspond to the outputs of the FSM can be obtained by a set of mappings $F = \{f_o, f_1, \ldots\}$ where $\forall x \in I$, $\exists f_i: C \to Z$ to emulate Mealy machines. If Moore machines are to be emulated, $F = \{f_o\}$, $f_o: C \to Z$.

Example 8.3.2 Consider the following FSM.

q \ x	0	1
A	A/0	B/1
B	B/0	A/0

The space S_4^1 is employed. Since the FSM has two states, A,B, two (4,2) cyclic strings are required. Let:

$$d_1 = a\ a\ b\ b$$

$$d_2 = c\ d\ d\ c$$

where:

$$h(d_1) = A$$

$$h(d_2) = B$$

Since the FSM has two inputs, there must be two transformations in TA where:

$$\tau_o(d_1) = d_1 \qquad \tau_1(d_1) = d_2$$

$$\tau_o(d_2) = d_2 \qquad \tau_1(d_2) = d_1$$

This follows from consideration of the next state for FSM under the two inputs, 0,1. The transformations τ_o, τ_1 are implied by σ_o and σ_1, respectively, where,

p	$\sigma_o(p)$	$\sigma_1(p)$
a a b	a	d
a b b	b	d
b b a	b	c
b a a	a	c
c d d	d	a
d d c	d	b
d c c	c	b
c c d	c	a
all others	Q	Q

The output functions for TA, f_o, f_1 can be written.

p	$f_o(p)$	$f_1(p)$
a a b b	0	1
c d d c	0	0

If TA is constructed, it will appear as shown in Figure 8.3.2.

The automaton constructed in the example can diagnose faulty state transitions but in no way detects faulty outputs made by the logic in F. The outputs made by TA can be diagnosed if the input set for TA is I x Z.

Example 8.3.3 Consider the previous example. Let the output z be fed back to each cell as shown in Figure 8.3.3. Since there are two inputs, $I = \{0,1\}$ and two outputs, $Z = \{0,1\}$, then there are four global transformations required, τ_{00}, τ_{01}, τ_{10}, τ_{11}, where for τ_{xy}, $x \in I$, $y \in Z$. The question arises as to what action should be taken if the output function, F, fails. We choose to place all cells in state Q for this

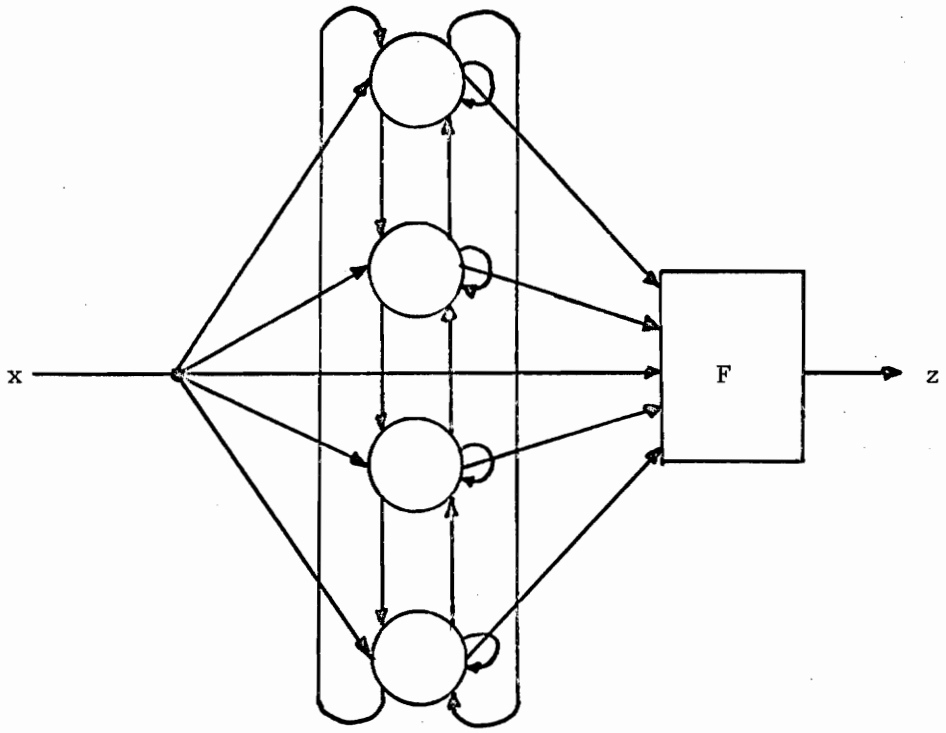


Fig. 8.3.2 A self-diagnosing automaton with Mealy output

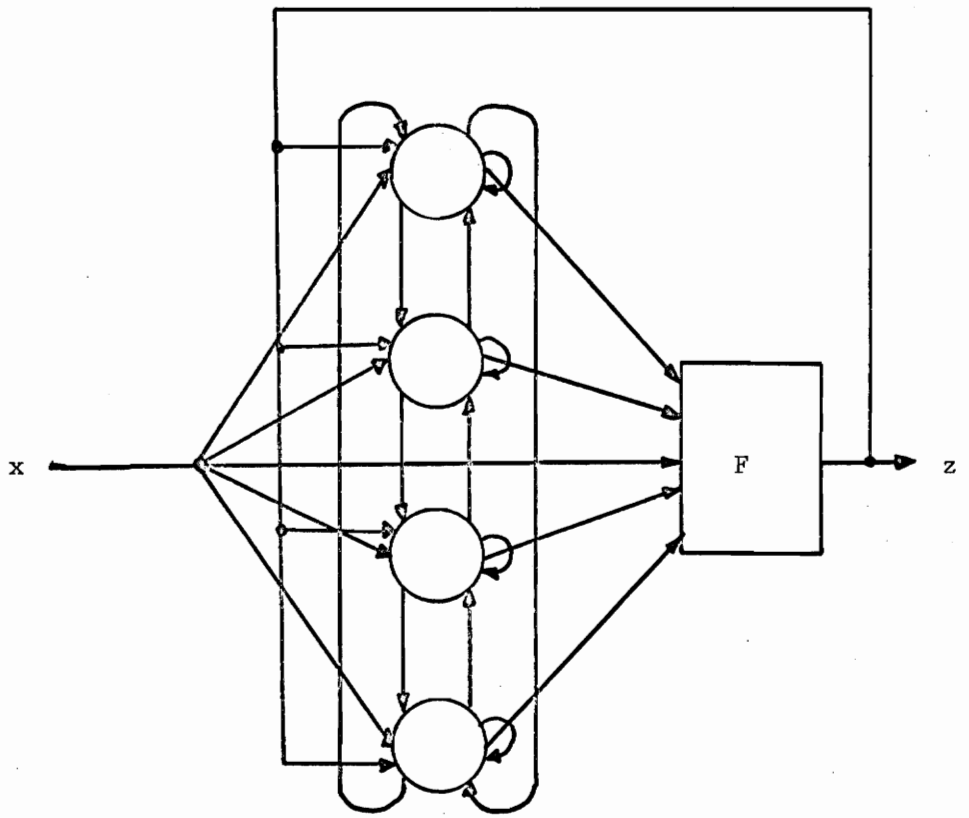


Fig. 8.3.3 A self-diagnosing automaton which detects output faults

example and defer discussion of this decision. For the example, the output of the FSM is:

q \ x	0	1
A	0	1
B	0	0

This implies that in TA,

$$\tau_{00}(d_1) = d_1 \qquad \tau_{10}(d_1) = QQQQ$$

$$\tau_{00}(d_2) = d_2 \qquad \tau_{10}(d_2) = d_1$$

$$\tau_{01}(d_1) = QQQQ \qquad \tau_{11}(d_1) = d_2$$

$$\tau_{01}(d_2) = QQQQ \qquad \tau_{11}(d_2) = QQQQ$$

This follows from the fact that:

q	x	z
A	0	1
A	1	0
B	0	1
B	1	1

are invalid state-input-output triples for FSM. It should be clear that TA will diagnose all state transition and output errors under the given global transformations. For each global transformation, there is a corresponding local transformation. These are defined as shown.

p	$\sigma_{00}(p)$	$\sigma_{01}(p)$	$\sigma_{10}(p)$	$\sigma_{11}(p)$
a a b	a	Q	Q	d
a b b	b	Q	Q	d
b b a	b	Q	Q	c
b a a	a	Q	Q	c
c d d	d	Q	a	Q
d d c	d	Q	b	Q
d c c	c	Q	b	Q
c c d	c	Q	a	Q
all others	Q	Q	Q	Q

The "splitting" of valid states in σ_{10}, σ_{11} is due to the fact that if the FSM is in state A, (d_1 for TA), then x = 1, z = 0 is invalid. Similarly, if the FSM is in state B, (d_2 for TA), then x = 1, z = 1 is invalid.

Corollary 8.3.5 $\forall$ FSM and $\forall k > 1$, $\exists$ a homomorphic TA = (A, S^1_{k2}, (-1,0, 1), I x Z) which is self-diagnosing and detects output faults.

Proof: If the input set to TA is I x Z (I and Z refer to the FSM), then $\forall(i,z) \in I \times Z$, $\exists\ \tau_{(i,z)}$ in TA. Recall that ω: Q x I $\rightarrow$ Z so $\omega(q,i) = z$. If h(d) = q, then let $\forall z^* \neq z$, $\tau_{(i,z^*)}(d) = \bar{Q}$ and $\tau_{(i,z)}(d) = h^{-1}[\delta(q)]$. (Recall that h is an arbitrary bijective map so h^{-1} exists on D.) Clearly, σ exists for the given τ so that TA detects output faults and is self-diagnosing.

As pointed out in the previous example, some consideration should be given to the desired action of TA if an output fault occurs. It may be possible that F will work correctly if TA reconfigures to an "equivalent" but different configuration. If the automaton goes to QQQQ, as in the example, the current state information will be lost. However, it is clearly possible to define special states Q_A, Q_B such that TA goes to $Q_AQ_AQ_AQ_A$ if an output fault occurs when TA is in d_1 or $Q_BQ_BQ_BQ_B$ if in d_2. This permits TA to retain the current state information. The symbol, Q, should be retained to diagnose faulty state transitions by cells. In terms of the previous example, the four local transformations would be:

p	$\sigma_{00}(p)$	$\sigma_{01}(p)$	$\sigma_{10}(p)$	$\sigma_{11}(p)$
a a b	a	Q_A	Q_A	d
a b b	b	Q_A	Q_A	d
b b a	b	Q_A	Q_A	c
b a a	a	Q_A	Q_A	c
c d d	d	Q_B	a	Q_B
d d c	d	Q_B	b	Q_B
d c c	c	Q_B	b	Q_B
c c d	c	Q_B	a	Q_B
all others	Q	Q	Q	Q

This permits separate diagnosis of state transitions and output faults and should aid in reconfiguration of TA.

The implications of Corollary 8.3.5 are three-fold. First, it demonstrates a parallel decomposition for finite-state machines where the resulting decomposition has a regular interconnection between identical machines. Second, the resulting decomposition is capable of self-diagnosis for state transitions of each cell and for output faults. Third, self-reconfiguration of the decomposition should be possible which will make it fault tolerant. The fault that the length of TA is k^2 where k is arbitrary permits larger decompositions. This will permit the existence of a large number of "equivalent" configurations and allow greater fault tolerance. Perhaps the most exciting aspect of this discovery is the possibility of a cellular array where each cell in the array looks like Figure 8.3.3. Such a construction could have self-reconfiguration within each cell and self-reconfiguration within the array. This fact will be given further consideration in the following chapter.

The following result is an extension of Lemma 8.3.1. It demonstrates the existence of homomorphic automata having different (arbitrary) dimensionality from its homomorphic image.

Theorem 8.3.6 $\forall TA = (A, E_L^d, X, \sigma)$, $\forall e > 1$, $\exists$ a homomorphic TA, $TA_h = (A_h, S_{L_h}^e, X_M, \sigma_h)$ which is self-diagnosing.

Proof: For TA_h, cyclic $(k^2,2)_e$ patterns are to be used as configurations. Construct such a pattern and let $S_{L_h}^e$ be its associated cyclic tessellation space. Clearly, as many cyclic $(k^2,2)_e$ patterns as are required can be created by replacing the symbols in the original pattern with a new alphabet. The new patterns will still be cyclic in $S_{L_h}^e$. It is clear that the construction of Lemma 8.3.1 will yield a homomorphic automaton. Thus, only self-diagnosis need be demonstrated. Consider the perturbation of a cell in some configuration in D. For a Moore neighborhood, X_M, there are 3^e neighbors for any cell, one of which is itself since $\xi_o \in X_M$ (see Figure 2.1.) The 3^e-1 neighbors of the perturbed cell still contain a $\overline{2}(e)$ subpattern which is unique $\forall d \in D$ but since each contains the perturbed cell at some location in their neighborhood, the resulting neighborhood configuration is not in any $d \in D$. Then these cells will go to state Q in c' since $\sigma(p) = Q$, $\forall p \ni \forall d \in D$, $p \notin d$. Then, the resulting automaton is self-diagnosing.

The following results are immediate extensions of Theorem 8.3.6.

Corollary 8.3.7 $\forall$ TA = (A, E_L^d, X, I), $\exists$ a homomorphic TA, $TA_h = (A_h, S_{L_h}^e, X_M, I_h)$ which is self-diagnosing.

Corollary 8.3.8 $\forall$ FSM, $\exists$ a homomorphic TA, $TA_h = (A_h, S^d_L, X_M, I \times Z)$ which is self-diagnosing and detects output faults.

Corollary 8.3.9 $\forall TA = (A, S^d_L, X, I)$, $\exists$ a homomorphic TA, $TA_h = (A_h, S^e_{L_h}, X_M, I)$ which is self-diagnosing.

The advantage to be gained by building multidimensional automata that simulate finite state machines is apparent when self-reconfiguration is considered. If more dimensions are available, certainly there will be more "equivalent" configurations available for self-reconfiguration. Furthermore, it should be possible for the array to "work its way around" faulty cells.

8.4 Reduction in Alphabet

Lemma 8.3.1 demonstrates a technique for creating a self-diagnosing TA which is homomorphic to a given TA. In doing so, a $(k^2,2)$ cyclic string using a unique alphabet was written for each and every configuration of the given TA. If the given TA is in a cyclic space which can support $(k^2,2)$ cyclic strings (for some k), then a reduction in $|A_h|$ is possible.

Prior to consideration of this possiblity, a special property of tessellation automata in any cyclic space is proved.

Theorem 8.4.1 $\forall TA = (A, S^d_L, X, \sigma)$, if $\tau(a) = b$, $a, b \in C$, then $\tau(\pi a) = \pi b$ where π is any element of a cyclic permutation group.

Proof: Consider a, b, πa, πb. Since π is an element of a cyclic permutation group, $\exists\, j \ni \forall i$, $\pi a(i+j) = a(i)$, $\pi b(i+j) = b(i)$. Then, $\pi a(N(X,i+j)) =$

$a(N(X,i))$ and $\pi b(N(X, i + j)) = b(N(X,i))$, $\forall i$. Since $\tau(a) = b$, then $\forall i$, $\sigma(a(N(X,i))) = b(i)$. Therefore, $\forall i$, $\sigma(\pi a(N(X,i + j))) = b(i)$. Let $c = \tau(\pi a)$. Then, $c(i + j) = b(i)$. Therefore, $c = \pi b$ and $\tau(\pi a) = \pi b$.

The following example is a special case of Corollary 8.3.9. The construction demonstrated permits a reduction in the cardinality of A_h if the original automaton is in a cyclic space for which $(k^2,2)$ cyclic strings exist.

Example 8.4.1 Given the following TA: $A = \{0,1\}$, S_4^1, $X = (0,1)$,

σ: 00 - 0

01 - 1

10 - 1

11 - 0

construct a homomorphic automaton, TA_h, which is self-diagnosing. Notice that TA is in S_4^1 which can support cyclic (4,2) strings. The global transformation, τ, for TA is found next.

i	c_i	$c_j=\tau(c_i)$	j
0	0 0 0 0	0 0 0 0	0
1	0 0 0 1	0 0 1 1	3
2	0 0 1 0	0 1 1 0	6
3	0 0 1 1	0 1 0 1	5
4	0 1 0 0	1 1 0 0	12
5	0 1 0 1	1 1 1 1	15
6	0 1 1 0	1 0 1 0	10
7	0 1 1 1	1 0 0 1	9
8	1 0 0 0	1 0 0 1	9
9	1 0 0 1	1 0 1 0	10
10	1 0 1 0	1 1 1 1	15
11	1 0 1 1	1 1 0 0	12
12	1 1 0 0	0 1 0 1	5
13	1 1 0 1	0 1 1 0	6
14	1 1 1 0	0 0 1 1	3
15	1 1 1 1	0 0 0 0	0

Consider the set of configurations, C, for TA when operated on by a cyclic permutation group, P, $P = (\pi, \pi^2, \pi^3, \pi^4)$ generated by $\pi = \begin{pmatrix} a & b & c & d \\ b & c & d & a \end{pmatrix}$. By Burnside's Theorem [31], P partitions C into N equivalence classes. The details of this follow.

$$N = \frac{1}{|P|} \sum_{\forall \pi^k \in P} F(\pi^k)$$

For our case, $|P| = 4$.

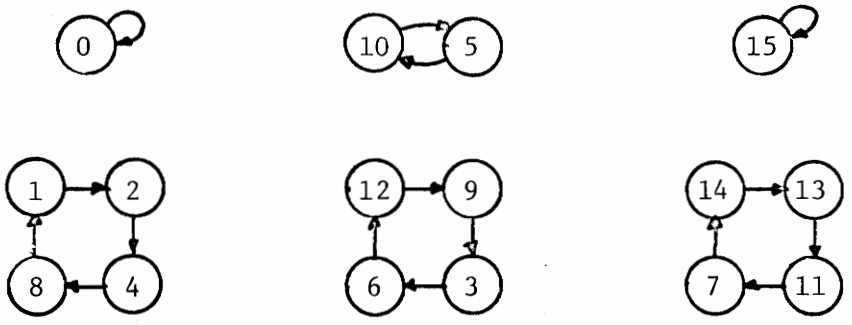
$F(\pi) = 2$ {0000,1111}

$F(\pi^2) = 4$ {0000,1111,0101,1010}

$F(\pi^3) = 2$ {0000,1111}

$F(\pi^4) = 16$ {all $c_i \in C$}

The sets indicated are the configurations which are invariant for the particular π^k. Then, $N = \frac{1}{4}(2 + 4 + 2 + 16) = 6$. Then there are six equivalence classes in C under P. Each class is called an orbit. These are constructed next. For simplicity, the configurations will be referred to by their subscript, i.



These will be referred to as the set E, $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ where $\forall e_i \in E$, e_i is a set. Then E is a set of sets. For the example,

$$e_1 = \{0\}$$

$$e_2 = \{5,10\}$$

$$e_3 = \{15\}$$

$$e_4 = \{1,2,4,8\}$$

$$e_5 = \{3,6,9,12\}$$

$$e_6 = \{7,11,13,14\}$$

For each set $e_i \in E$, select an element, $\hat{e}_i$. For instance,

$$\hat{e}_1 = 0$$

$$\hat{e}_2 = 5$$

$$\hat{e}_3 = 15$$

$$\hat{e}_4 = 1$$

$$\hat{e}_5 = 3$$

$$\hat{e}_6 = 7$$

Also, let $\hat{E}$ be the collection of $\hat{e}_i$. Then, $\hat{E} = \{0, 1, 3, 5, 7, 15\}$. Then $\hat{E}$ is the set of coset leaders and, if P is applied to $\hat{E}$, then the orbits of C are generated. Notice that, by Theorem 8.4.1, if $\tau(\hat{e}_i) \in e_j$, then $\forall c_k \in e_i$, $\tau(c_k) \in e_j$. That is, all elements of an equivalence class map to the same equivalence class under τ. Now, TA_h is constructed. For each $\hat{e}$ in $\hat{E}$, construct a (4,2) cyclic string to be used as a configuration in TA_h. Let each string employ a unique alphabet and refer to the collection of these configurations as $\hat{E}_h$. For the example, six such strings are required. Let:

$$\hat{e}_{0h} = A\ A\ B\ B \qquad \hat{e}_{5h} = G\ G\ H\ H$$

$$\hat{e}_{1h} = C\ C\ D\ D \qquad \hat{e}_{7h} = I\ I\ J\ J$$

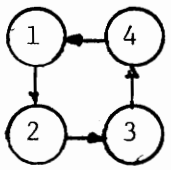
$$\hat{e}_{3h} = E\ E\ F\ F \qquad \hat{e}_{15h} = K\ K\ L\ L$$

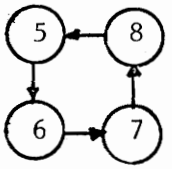
which is consistent with $\hat{E}$ for TA. This new set, $\hat{E}_h$, is the generator of the equivalence classes of the subset of configurations, D, in TA_h. Next, a mapping, g, g: $A_h^3 \rightarrow A$ is defined.

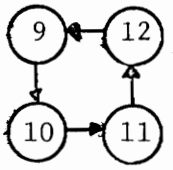
g:	A A B - 0	G G H - 1
	A B B - 0	G H H - 0
	B B A - 0	H H G - 1
	B A A - 0	H G G - 0
	C C D - 0	I I J - 1
	C D D - 0	I J J - 1
	D D C - 1	J J I - 1
	D C C - 0	J I I - 0
	E E F - 0	K K L - 1
	E F F - 1	K L L - 1
	F F E - 1	L L K - 1
	F E E - 0	L K K - 1

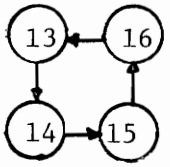
If g is applied to the elements of $\hat{E}_h$, an element of $\hat{E}$ is found. (This is done in the same fashion that σ is applied to find the next configuration.) Next, P is applied to $\hat{E}_h$ to find D, the diagnosable configurations of TA_h and the orbits of D under P are constructed.

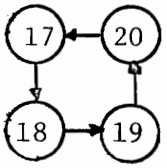
i	d_i	i	d_i	i	d_i
1	A A B B	9	E E F F	17	I I J J
2	A B B A	10	E F F E	18	I J J I
3	B B A A	11	F F E E	19	J J I I
4	B A A B	12	F E E F	20	J I I J
5	C C D D	13	G G H H	21	K K L L
6	C D D C	14	G H H G	22	K L L K
7	D D C C	15	H H G G	23	L L K K
8	D C C D	16	H G G H	24	L K K L

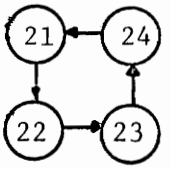












This should come as no surprise since the strings have disjoint alphabets and are of length four. Next, g is applied to each element of D. This defines a second mapping h: $D \rightarrow C$. For simplicity, only the index of the configurations is used.

d_i	$h(d_i)$	d_i	$h(d_i)$
1	0	13	5
2	0	14	10
3	0	15	5
4	0	16	10
5	1	17	7
6	2	18	14
7	4	19	13
8	8	20	11
9	3	21	15
10	6	22	15
11	12	23	15
12	9	24	15

Next, τ_h is defined where $\tau_h(d_i) = d_j$ if and only if $\tau[h(d_i)] = h(d_j)$. This is done only for the elements of $\hat{E}_h$. For the example,

$d_i \in \hat{E}_h$	$h(d_i)$	$\tau[h(d_i)]$	$\tau_h(d_i)$
1	0	0	2 (1,3,4)
5	1	3	9
9	3	5	13 (15)
13	5	15	21 (22,23,24)
17	7	9	12
21	15	0	4 (1,2,3)

Notice that more than one choice for $\tau_h(d_i)$ is available. The alternate choices are those placed in parenthesis in the final column. The choice is arbitrary so long as $h[\tau_h(d_i)] = \tau[h(d_i)]$. The purpose of this step is to force τ_h to move TA_h between equivalence classes (in D) in the same way that τ moves TA. Since all configurations of $\hat{E}_h$ are (4,2) cyclic strings, σ exists for the chosen τ. The mapping g is also shown.

$p \in A_h^3$	$\sigma(p)$	$g(p)$
A A B	B	0
A B B	B	0
B B A	A	0
B A A	A	0
C C D	E	0
C D D	F	0
D D C	F	1
D C C	E	0
E E F	G	0
E F F	H	1
F F E	H	1
F E E	G	0
G G H	K	1
G H H	L	0
H H G	L	1
H G G	K	0
I I J	E	1
I J J	E	1
J J I	F	1
J I I	F	0
K K L	A	1
K L L	A	1
L L K	B	1
L K K	B	1

This completes the construction of TA_h. To verify the construction, τ_h is calculated for each $d_i \in D$.

i	d_i	$d_j=\tau(d_i)$	j
1	A A B B	A B B A	2
2	A B B A	B B A A	3
3	B B A A	B A A B	4
4	B A A B	A A B B	1
5	C C D D	E E F F	9
6	C D D C	E F F E	10
7	D D C C	F F E E	11
8	D C C D	F E E F	12
9	E E F F	G G H H	13
10	E F F E	G H H G	14
11	F F E E	H H G G	15
12	F E E F	H G G H	16
13	G G H H	K K L L	21
14	G H H G	K L L K	22
15	H H G G	L L K K	23
16	H G G H	L K K L	24
17	I I J J	F E E F	12
18	I J J I	E E F F	9
19	J J I I	E F F E	10
20	J I I J	F F E E	11
21	K K L L	B A A B	4
22	K L L K	A A B B	1
23	L L K K	A B B A	2
24	L K K L	B B A A	3

The homomorphism can be verified.

i	$h(d_i)$	$\tau[h(d_i)]$	$\tau_h(d_i)$	$h[\tau_h(d_i)]$
1	0	0	2	0
2	0	0	3	0
3	0	0	4	0
4	0	0	1	0
5	1	3	9	3
6	2	6	10	6
7	4	12	11	12
8	8	9	12	9
9	3	5	13	5
10	6	10	14	10
11	12	5	15	5
12	9	10	16	10
13	5	15	21	15
14	10	15	22	15
15	5	15	23	15
16	10	15	24	15
17	7	9	12	9
18	14	3	9	3
19	13	6	10	6
20	11	12	11	12

i	$h(d_i)$	$\tau[h(d_i)]$	$\tau_h(d_i)$	$h[\tau_h(d_i)]$, cont.
21	15	0	4	0
22	15	0	1	0
23	15	0	2	0
24	15	0	3	0

As is seen, $\forall i$, $\tau[h(d_i)] = h[\tau_h(d_i)]$. Then TA and TA_h are homomorphic. Since TA_h has (4,2) cyclic strings as configurations, it follows that, if all remaining $p \in A_h^3$ have $\sigma(p) = Q$, then TA_h is self-diagnosing.

It should be clear to the informed reader that such a construction will always exist if the original automaton is in a cyclic space which can support cyclic $(k^2,2)_d$ patterns. The only obscurity is that the orbits of D under P may not have the same cardinality as the orbits of C under P. However, the cardinality of any orbit of C must divide the cardinality of any orbit of D. This fact follows as a corollary to Burnside's Theorem [31]. Since π generates the orbits of both, we see that g exists so that $\tau[h(d_i)] = h[\tau_h(d_i)]$.

Since this example is a special case of Corollary 8.3.9, no formalization will be produced. The construction demonstrated merely reduces A_h from the construction given in Lemma 8.3.1. Furthermore, it will not reduce $|A_h|$ when applied to finite-state machines unless the FSM is not reduced. If the given FSM has equivalent states, this construction could be employed, but it should be clear that no improvement in $|A_h|$ will be realized if the machine is reduced before Lemma 8.3.4 is applied. It should be equally clear that it cannot be applied to automata which are in E_L^d or E^d since "rotation" has no meaning for these spaces. Perhaps a similar technique can be found for these spaces if "shifting" is considered. Certainly a result identical to Theorem 8.4.1 exists,

but Burnside's Theorem cannot be employed. If the space of the original automaton is cyclic but does not support $(k^2,2)$ cyclic strings, then the new automaton must be in a different space since it must employ $(k^2,2)$ cyclic strings as configurations. Then two different permutation groups are involved. A technique which considers this case may exist but it will not be demonstrated here.

8.5 Discussion

The results of this chapter demonstrate the feasibility of self-diagnosis in finite tessellation automata. These results should pave the way for automata which are highly reliable. Once a formalization of configuration equivalence is made, a study of self-reconfiguration following self-diagnosis should be feasible. Of the results presented in this chapter, certainly Corollary 8.3.5 is the most significant. The construction of Example 8.3.4 could prove of some use for some special cases of arrays. Furthermore, self-diagnosis during synthesis is clearly possible if each configuration in the synthesis process is a $(k^2,2)_d$ pattern. This will raise the bounds found in Chapter VII somewhat but is certainly worth the extra symbols.

IX. SUMMARY

First, we wish to discuss briefly the chapters which precede these remarks and place them in perspective. Chapter I serves to introduce cellular automata and provides a brief historical account of their development. In Chapter II, the formal notion of tessellation automata is advanced and previous research efforts concerning these automata are summarized. An architecture which is potentially self-diagnosable and self-reconfigurable is introduced in Chapter III. The proposal is discussed and research which affected the final model is summarized. The proposed architecture is composed of a tessellation automaton and a cellular array. The tessellation automaton is shown to be computationally equivalent to Turing machines in Chapter IV. This demonstrates the powerful nature of the tessellation automaton. Also considered is the stability question for these automata. Chapter V introduces and solves an interesting combinatorial problem concerning patterns which is an extension of work by I. J. Good. Chapter VI considers the growth rate of configurations in various tessellation automata. The results of Chapter IV and V are combined in Chapter VII, which demonstrates a technique for generating particular configurations in various tessellation automata. In Chapter VIII, results regarding perturbation of the cells within the tessellation automaton are advanced. Of greatest interest is the simulation of finite state machines by tessellation automata which can detect any cell or output fault. These results are independent of the fault type due to the fact that the detection of faults is done at the functional level rather than at the circuit level.

Next, we wish to list areas of further study and to indicate the problems presented by some. As for the specific areas presented within this dissertation, most still represent fertile areas for research since most of the results found are sufficient but not necessary. This fact follows, primarily, from the "constructive" nature of automata theory. Since the results presented "construct" various automata to do various jobs, it cannot be claimed that it is the only construction which does so. Perhaps the given constructions are "best" (they work) but a "better" (in some sense) construction may exist. (There is more than one way to skin an automaton.) For this reason, further research to find a "better" (or at least, different) way should be encouraged. Aside from this, several extensions seem possible and some unaddressed questions exist.

(1) Equivalent configuratons.

(2) Self-reconfiguration.

(3) Self-diagnosis of multiple perturbations.

(4) Patterns which are maximal.

(5) Patterns which are cyclic in the "usual" sense for all dimensions.

(6) Patterns with non-contiguous subpatterns.

(7) Patterns which are non-hypercubic.

(8) Applications of patterns as codes.

Of the areas indicated, the finding of equivalent configurations is the most pressing. Certainly, Polya's Theorem permits counting of configurations which are equivalent under rotation but this is not the type of equivalence that is desired. In terms of Example 8.3.1, it is desired

to find a set of configurations which have the same I/O characteristics and state transitions as those given, but which do so with quarantined cells in the configuration. Implicit in this statement is that the output function, F, has certain properties. Clearly, the question of equivalence is a difficult question. Once solved, self-reconfiguration should be straightforward since getting the automaton to move from a perturbed configuration which has been diagnosed to an equivalent configuration should be relatively easy.

Another important question would be the extension of results found in Chapter VIII to diagnose multiple perturbations. It should be clear that the results presented will diagnose all perturbed cells which are separated by a distance of four or more in the automaton. This is simply n+1 where n is the scope of X and, for the results presented, n=3. It seems possible that multiple perturbations within the scope of X might be diagnosable if X is enlarged while the configuration remains a $(k^2,2)_d$ pattern. Certainly, this deserves investigation.

The questions listed concerning patterns may arise when configuration equivalence is studied. It should be obvious that nearly all the results found in this dissertation are founded on the results for patterns of Chapter V. The extensions listed are problems which did not require an answer for our purposes but may arise in some other effort to come. These are all difficult questions.

If the patterns of Chapter V are considered as codes, they permit detection of all single errors and some multiple errors. Since the symbol in error can be pointed out (diagnosed), error correction should be possible. It is clear that if the alphabet for the pattern is binary,

all single errors can be corrected. (It is known which symbol is in error and there is only one possible alternate choice for it.) If more symbols are permitted, iteration could be used to find the correct symbol and thereby correct the error. All of this could be done in tessellation automata with a cyclic space. If the code is received serially, such an iteration could be very useful due to the "parallel" nature of the "correcting" automaton since quite a lot of time could elapse between received codes.

In closing, some remarks concerning the application of the results in Chapter VIII are in order. The machine of Figure 8.3.2 employs a single logic block, F, for computation of z, the output. If the machine is to be completely cellular, this logic block must be decentralized. There appear to be at least two ways to decentralize F. One way is to make F redundant. That is, place identical output modules with each cell. Figure 9.1 demonstrates this construction. By feeding back all of the outputs made (by each F module), faulty output detection is possible. This construction has the advantage that the output made by the array is available in several locations so if an output module fails, others are available. A clearcut disadvantage is the fact that all output modules must receive the state information from all cells to compute z. This implies a tremendous number of interconnections between the control structure (the cells) and the computation structure (the F modules). A second possibility exists and is shown in Figure 9.2. Here, the output, z, is decomposed into several modules, F_1 through F_4, which receive the state information from the cells adjacent (under X) to the

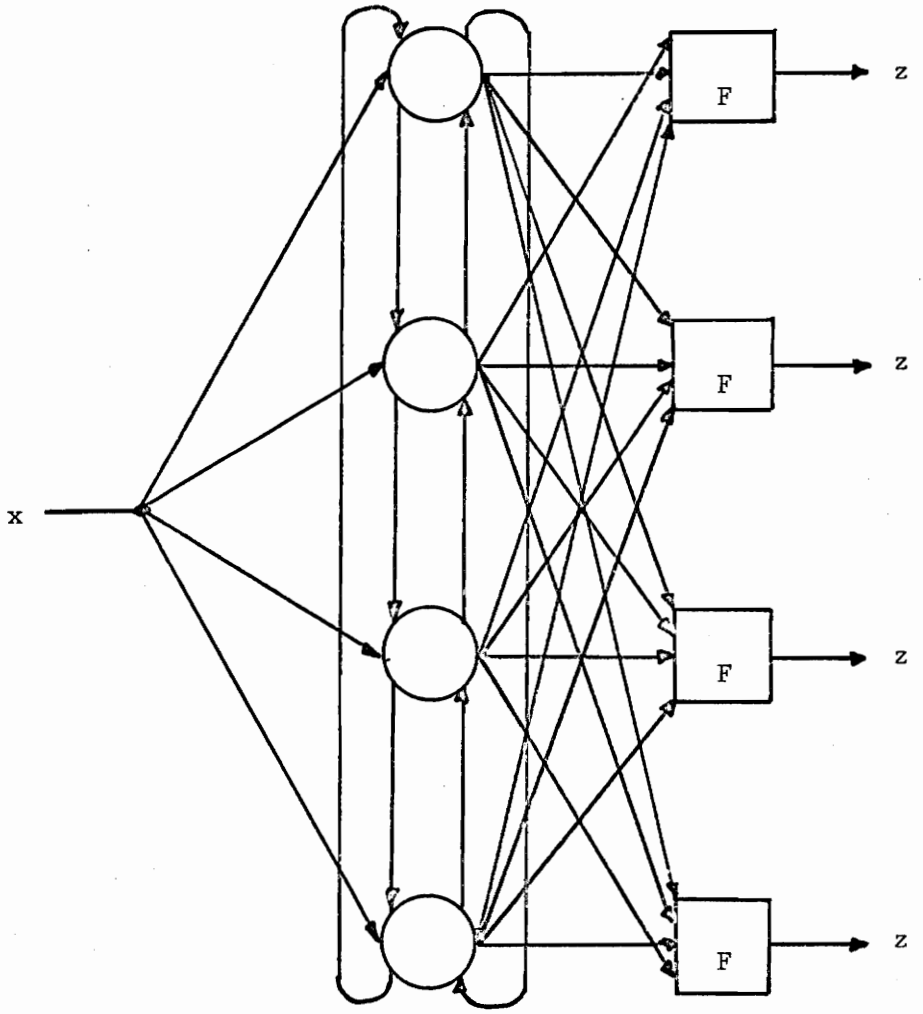


Fig. 9.1 Redundant Modules for Cellular Output

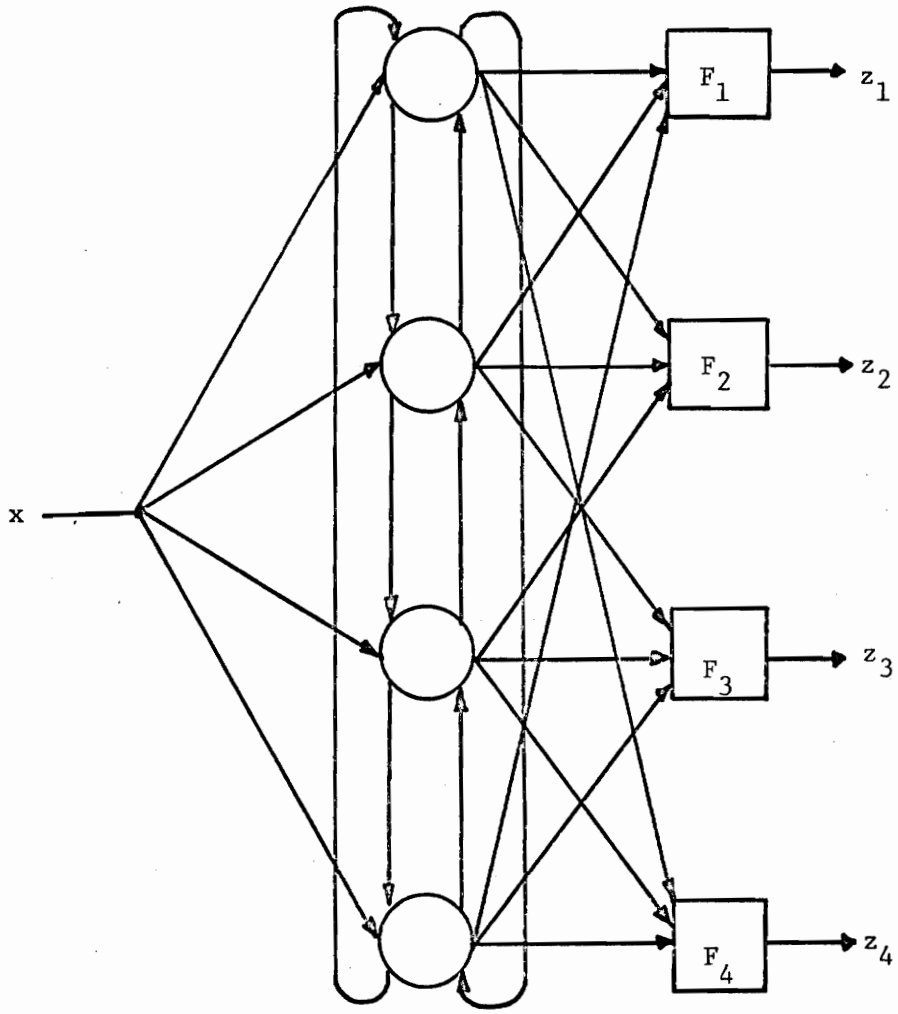


Fig. 9.2 Localized Output Modules

F module's cell. We refer to this as localized output. The outputs, z_1 through z_4, make up the array output, z. These localized outputs could be binary in nature. For instance, if z = 0011 is required for some state-input combination, then z_1=0, z_2=0, z_3=1, z_4=1 would yield the required output. In this case, the realization is a sort of "bit slice" of the output. Obviously, other "n-bit slice" decompositions of z are possible. To detect output faults, all outputs could be fed back to all cells. This requires a large number of interconnections and defeats the purpose of using local output modules. Fortunately, this is not necessary. If the output from each local module is fed back only to its associated cell, the local transformation can be defined so that if the output is in error, the cell goes to a "wrong" state (probably Q is the best choice for this). Then, the cell's neighbors will quarantine it at the next step. The construction of such an array is shown in FIgure 9.3. If the neighborhood configurations are unique in each global configuration of the cyclic tessellation automaton, then a single module, G, can be defined to produce the outputs, z, through z_r. Clearly, further research dealing with the particular details of this construction are in order.

As a final question, a comparison between tessellation automata and iterative arrays is suggested. Both of these are cellular in nature and appear equivalent at the outset. However, a difficulty exists if the arrays are infinite. If the cardinality of the tessellation automaton's alphabet is denoted by k, ($|A| = k$), then there are exactly $k^{(k^n)}$ global transformations in the input set for TA. (Recall that $|X| = n$.) The

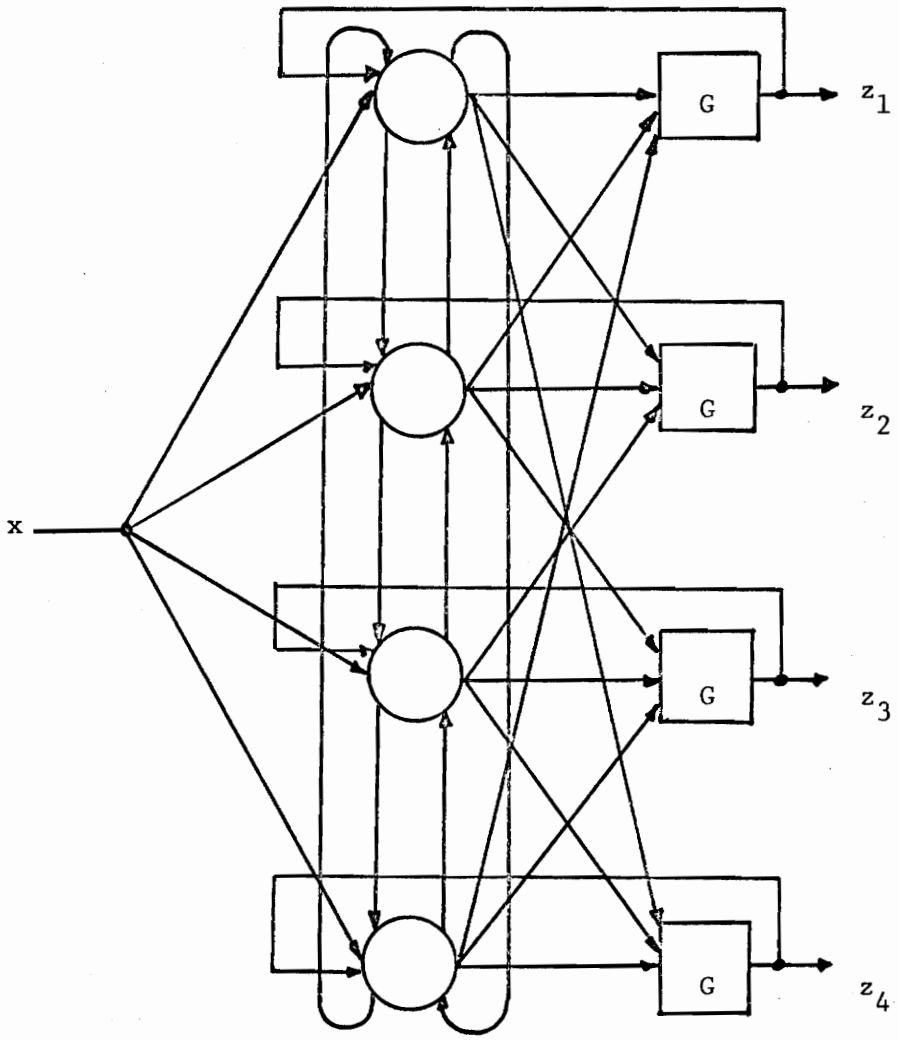


Fig. 9.3 Localized Output with Fault Detection

iterative array has an infinite number of inputs since there is an input made to each cell. This is the difficulty. If these two automata are to be equivalent in any sense, there must exist a finite number of partitions of the input set for the iterative array where every element in a partition causes the same next state (globally) for any given current state (global). This is an interesting question which we leave for future effort.

BIBLIOGRAPHY

[1] S. Amoroso and G. Cooper, "Tessellation Structures for Reproduction of Arbitrary Patterns," J. Comput. Sys. Sci., 5, 455-464, 1971.

[2] S. Amoroso and Y. N. Patt, "Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures, J. Comput. Sys. Sci., 6, pp. 448-464, 1972.

[3] S. Amoroso and R. Guilfoyle, "Some Comments on Neighborhood Size for Tessellation Automata", Information and Control, 21, pp. 48-55, 1972.

[4] S. Amoroso and I. J. Epstein, "Maps Preserving the Uniformity of Neighborhood Interconnection Patterns in Tessellation Structures, Information and Control", 25, pp. 1-9, 1974.

[5] M. A. Arbib, Theories of Abstract Automata, Prentice-Hall, Inc., Englewood, N.J. (1969).

[6] A. W. Burks, Essays on Cellular Automata, University of Illinois Press, Urbana, IL., 1970.

[7] S. Cole, "Real-Time Computation by N-dimensional Arrays of Finite-State Machines", IEEE Conf. Rec. 7th Ann. Switching and Autom. Theory, pp. 53-77, 1966.

[8] A. Church, "An Unsolvable Problem of Elementary Number Theory", Amer. J. Math., 58, pp. 345-363, 1936.

[9] W. L. Gregory, "Test Scheduling and Configuration in A Self-Repairing Computer", M.S. Thesis, VPI & SU, 1977.

[10] F. C. Hennie, Finite-State Models for Logical Machines, John Wiley and Sons, N.Y. 1968.

[11] F. C. Hennie, Iterative Arrays of Logical Circuits, John Wiley & Sons, N.Y., 1961.

[12] F. C. Hennie, "Analysis of Bilateral Iterative Networks", IRE Trans. Circuit Theory, CT-6, pp. 35-45, 1959.

[13] J. H. Holland, Essays on Cellular Automata, (Edited by A. W. Burks) University of Illinois Press, 1970.

[14] J. E. Hopcroft and J. D. Ullman, Formal Languages and Their Relation to Automata, Addison-Wesley Publishing Co., Reading, Mass., 1969.

[15] C. L. Liu, Introduction to Combinatorial Mathematics, McGraw-Hill Book Co., N.Y., 1968.

[16] E. F. Moore, "Machine Models of Self-Reproduction", Notices of the American Math. Soc., 6, pp. 622-623, 1959.

[17] J. Myhill, "The Converse of Moore's Garden-of-Eden Theorem", Proc. of the Amer. Math. Soc., 14, pp. 685-686, 1963.

[18] Y. N. Patt, "Injections of Neighborhood Size Three and Four on the Set of Configurations from the Infinite One-Dimensional Tessellation Automata of Two-State Cells", Proc. of the 1972 Southeastern Symposium on Systems Theory, pp. 123-127.

[19] E. Post, "A Variant of a Recursively Unsolvable Problem", Bull. Am. Math. Soc., 52, pp. 264-268, 1946.

[20] D. D. Stewart, "A Cellular Self-Repairing Cordic Structure", M.S. Thesis, VPI & SU, 1977.

[21] J. Thatcher, "The Construction of a Self-Describing Turing Machine", Proc. of the Symp. on Math. Theory of Automata, April 1962, pp. 165-171.

[22] R. A. Thompson, S. M. Walters and F. G. Gray, "Stability in A Class of Tessellation Automata", Proc. of the Ninth Annual Southeastern Symposium on System Theory, pp. 404-414, 1977.

[23] A. M. Turing, "On Computable Numbers with An Application to the Entscheidungs-problem", Proc. London Math. Soc., 2-42, pp. 230-265. A correction, ibid., 43, pp. 544-546, 1936.

[24] S. F. Unger, "Pattern Recognition Using Two-Dimensional Bilateral Interactive Combinational Switching Circuits", Proc. Symposium on Mathematical Theory of Automata, 1962.

[25] J. Von Neumann, Theory of Self-Reproducing Automata, University of Illinois Press, Urbana, IL., 1966.

[26] S. M. Walters, R. A. Thompson and F. G. Gray, "Pattern Synthesis in One-Dimensional Tessellation Automata", Proc. of the Eighth Annual Southeastern Symposium on System Theory, pp. 11-12, 1976.

[27] S. M. Walters, R. A. Thompson and F. G. Gray, "Pattern Restoration in Tessellation Automata", Proc. of Southeastcon '77, pp. 646-648.

[28] H. Yamada and S. Amoroso, "Tessellation Automata", Information and Control, 14, pp. 299-317, 1969.

[29] H. Yamada and S. Amoroso, "A Completeness Problem for Pattern Generation in Tessellation Automata", J. Comp. Sys. Sci., 4, pp. 137-176, 1970.

[30] H. Yamada and S. Amoroso, "Structural and Behavioral Equivalences of Tessellation Automata", Information and Control, 18, pp. 1-31, 1971.

[31] J. J. Rotman, The Theory of Groups, Allyn and Bacon, Inc., Boston.

[32] G. Birkhoff and T. C. Bartee, Modern Applied Algebra, McGraw-Hill, Inc., N.Y., 1970.

[33] I. J. Good, "Normal Recurring Decimals", J. London Math. Soc., vol. 21, pp. 167-169, 1946.

[34] M. Hall, Combinatorial Theory, Blaisdell Pub. Co., Waltham, Mass.

[35] M. Hoptiak, "Implementation of A Testing and Initialization Algorithm for Generalized Tree Structure", M.S. Thesis, VPI & SU, 1977.

[36] M. Hoptiak, F. G. Gray and R. A. Thompson, "Implementation of a Testing and Initialization Algorithm for A Generalized Tree Structure", Proc. of Ninth Southeastern Symposium on System Theory, 182-191, 1977.

[37] L. C. Shih, "Fault Detection and Location in Modular Trees", M.S. Thesis, VPI & SU, 1976.

[38] L. C. Shih and F. G. Gray, "Fault Location in Modular Trees", Proc. of Eighth Southeastern Symposium on System Theory, pp. 238-243, 1976.

[39] B. A. Prasad, "Multiple Fault Detection in Iterative Logic Structures", Ph.D. Dissertation, VPI & SU, 1974.

[40] B. A. Prasad and F. G. Gray, "Detection of Multiple Unrestricted Fault in Generalized Tree Structures", Proc. of the Sixth Annual Southeastern Symposium on System Theory, Baton Rouge, LA., February 1974.

[41] B. A. Prasad and F. G. Gray, "Fault Diagnosis in Uniform Modular Realizations of Sequential Machines", Digest of Papers, 1973 International Symposium on Fault Tolerant Computing, Palo Alto, CA., June 1973.

[42] B. A. Prasad and F. G. Gray, "Multiple Fault Detection in One-Dimensional Iterative Arrays", Proc. of the Fifth Annual Southeastern Symposium on System Theory, Raleigh, N.C., pp. 333, March 1973.

[43] A. D. Friedman, "Feedback in Synchronous Sequential Switching Circuits", IEEE Trans. on Electronic Computers, Vol. EC-15, pp. 354-367, June 1966.

[44] T. F. Arnold, C. J. Tan and M. M. Newborn, "Iteratively Realized Sequential Circuits", IEEE Trans. on Computers, Vol. C-19, pp. 54-66, January 1970.

[45] S. W. Golomb, Shift-Register Sequences, Holden-day, San Francisco, 1967.

[46] F. G. Gray and R. A. Thompson, "Reconfiguration for Repair in a Class of Universal Logic Modules", IEEE Trans. on Computers, Vol. C-23, pp. 1185-1194, November 1974.

[47] D. W. Nelson, "Advanced Universal Modular Tree Design Techniques", M.S. Thesis, VPI & SU, 1976.

[48] R. A. Thompson and F. G. Gray, "Universal Modular Trees: A Design Procedure", to appear in IEEE Trans. on Computers.

[49] R. A. Thompson and F. G. Gray, "Definite Acceptors and Universal Modular Trees", submitted to IEEE Trans. on Computers.

[50] J. S. Walther, "A Unified Algorithm for Elementary Functions", Proc. of Spring Joint Computer Conference, 1971, pp. 379-385.

[51] J. C. Fletcher, W. M. Daly and J. F. McKenna, "Fault Tolerant Clock Apparatus Utilizing a Controlled Minority of Clock Elements", U.S. Patent #3,900,741.

[52] S. M. Walters, R. A. Thompson and F. G. Gray, "A Fault Tolerant Computing Structure", submitted, 1978 International Symposium on Fault-Tolerant Computing.

[53] S. M. Walters, R. A. Thompson and F. G. Gray, "Extensions of the Teleprinter Problem", submitted, Journal of the London Math. Soc.

VITA

S. M. Walters was born on December 28, 1945, in Nashville, Tennessee, and received his high school diploma in June 1963, at Huntsville, Alabama. His education at Auburn University, Auburn, Alabama, was interrupted by military service, but he returned to Auburn and received the Bachelor of Electrical Engineering degree in August 1974. He joined the Electrical Engineering Department of Virginia Polytechnic Institute and State University and received the Master of Science degree in Electrical Engineering in June 1976. During the academic year 1976-77, he was an Instructor of Electrical Engineering at VPI & SU.

He is a member of Sigma Xi, Eta Kappa Nu, Phi Kappa Phi, IEEE, and received the USAF Commendation Award for meritorious service.

He has accepted a position with Bell Telephone Laboratories in Holmdel, New Jersey, following completion of the doctorate degree.

Stephen M. Walters

PATTERN SYNTHESIS AND PERTURBATION IN TESSELLATION AUTOMATA

by

S. M. Walters

(ABSTRACT)

Following a brief introduction to cellular automata, the formal notion of tessellation automata is advanced and a cellular computer using them is described. Potentially, this computer has the ability to do self-diagnosis and self-reconfiguration for internal faults without external assistance. Next, computational equivalence of Turing Machines and tessellation automata is demonstrated. This shows the powerful nature of tessellation automata. Following this, the existence of cyclic patterns with unique subpatterns is proven. These results are relied upon heavily for the remainder of this dissertation. Results which find the relation between growth rate and neighborhood index are then obtained and a solution to the synthesis problem is found. This demonstrates the existence of a local transformation whereby a given configuration can be generated from a single seed in the automaton. Next, results for single perturbations of the automaton are given and fall into two classes; restoration of equilibrium configurations and self-diagnosis. Restoration results demonstrate the existence of configurations which will immediately restore themselves following any single perturbation. Results obtained for self-diagnosis show that any single perturbation of certain configurations will be detected and the perturbed cell will be identified by its neighbors. Of particular interest is a result demonstrating the construction of a homomorphic self-diagnosing automa-

ton for any given finite-state machine. Implications of the results obtained are discussed and some open problems are considered.