Journal of Statistical Physics, Vol. 43, Nos. 1/2, 1986

Invariant Strings and Pattern-Recognizing Properties of One-Dimensional Cellular Automata

Erica Jen[1,*]

Received August 12, 1985

A cellular automaton is a discrete dynamical system whose evolution is governed by a deterministic rule involving local interactions. It is shown that given an arbitrary string of values and an arbitrary neighborhood size (representing the range of interaction), a simple procedure can be used to find the rules of that neighborhood size under which the string is invariant. The set of nearest-neighbor rules for which invariant strings exist is completely specified, as is the set of strings invariant under each such rule. For any automaton rule, an associated "filtering" rule is defined for which the only attractors are spatial sequences consisting of concatenations of invariant strings. A result is provided defining the rule of minimum neighborhood size for which an arbitrarily chosen string is the unique invariant string. The applications of filtering rules to pattern recognition problems are discussed.

KEY WORDS: Cellular automata; invariant strings; pattern recognition; filtering.

1. INTRODUCTION

A cellular automaton is a discrete mathematical system whose time evolution depends deterministically on local interactions. Originally introduced by von Neumann and Ulam[1] as potential models for biological self-reproduction, cellular automata have since been used as mathematical tools for studying a wide variety of problems. (See Ref. 2 for a collection of articles on the subject.) In general, a cellular automaton can be defined as a spatial lattice of sites whose values at each time step are determined as a function of the values of the "neighboring" sites at the

[1] Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, New Mexico 87545.

* Permanent address: Mathematics Department, University of Southern California, Los Angeles, CA 90089.

0022-4715/86/0400-0243$05.00/0

previous time step. The site values are restricted to a finite set of integers, and specification of the function provides the "rule" governing the automaton's behavior. Consider, for example, cellular automata defined on a one-dimensional lattice of sites $\{x_i, -\infty < i < \infty\}$, each of which can assume any of the values $V = \{0,..., k-1\}$. The general form of a rule for such an automaton is then given by

$$x_i^{t+1} = f(x_{i-r}^t,..., x_i^t,..., x_{i+r}^t) \qquad f: V^{2r+1} \to V \tag{1.1}$$

where $r \geqslant 0$ represents the size (or radius) of the neighborhood considered by the rule, and the initial condition is specified by the values $\{x_i^0; -\infty < i < \infty\}$. Note that the function f is chosen to be independent of t and x_i^t, and the site values are computed synchronously (in parallel) at each time step.

This paper will explore two topics related to the existence of "invariant strings" for cellular automata. An invariant string of an automaton rule is defined to be a finite spatial sequence of site values that remains invariant under the rule, independent of the sequence's spatial position or the values of its neighboring sites. The first topic to be considered is the existence of invariant strings for rules with a fixed neighborhood size. Results will be presented for elementary rules (radius r equal to 1), but the analysis could be extended in a straightforward, albeit increasingly more cumbersome, fashion to the case of any fixed neighborhood size. Specifically, it will be shown in Section 2 that given an arbitrary string of 0's and 1's, a simple procedure can be used to construct the elementary rules under which the string is invariant. Furthermore, it is possible to provide a complete characterization of the set of elementary automata rules for which invariant strings exist, together with the set of invariant strings associated with each such rule. In addition, for a given rule R, an associated "filtering" rule R^- (with the same radius) will be defined in Section 3 for which arbitrary initial conditions are mapped to spatial sequences consisting essentially of the invariant strings for R, and those invariant strings appear in the spatial sequences precisely where they did in the initial conditions.

The second topic to be considered is the identification of rules of unspecified neighborhood size for which a given string is invariant. In particular, Section 4 will characterize the automaton rule of minimal neighborhood size for which an arbitrarily chosen string represents the only invariant string. As the basis of this characterization, the concept of the "longest self-match" in a string will be introduced. The longest self-match is defined to be the longest tuple that occurs more than once in the string. The neighborhood size of the "minimal-radius" rule possessing the desired property will be shown to be determined by the length of the

longest self-match of the given string. Once the appropriate neighborhood size has been found, the rule itself is then easily constructed.

The results on the existence of invariant strings have direct implications for the use of cellular automata in pattern recognition. The problem of defining a rule for which a given string is invariant is equivalent to the problem of defining a rule that will "recognize" a specified pattern in an arbitrary input sequence. The characterization of all strings that are invariant under a rule R is equivalent to the characterization of all patterns that can be recognized by R. Furthermore, the extension of automata rules to "filtering" rules provides a mechanism by which specified patterns in an input sequence will be preserved, and all others (representing "noise") annihilated. In particular, the result on the minimum neighborhood size of the rule for which a given string constitutes the only invariant string provides the unique automaton that represents an optimally efficient filter for that pattern. Moreover, the number of steps needed to filter for the pattern is bounded above by a quantity depending only on the pattern itself, and thus is independent of the actual input data. In sum, the problems of pattern recognition provide a natural context in which the mathematical results on invariant strings for cellular automata can be usefully exploited.

2. INVARIANT STRINGS FOR ELEMENTARY CELLULAR AUTOMATA

"Elementary" cellular automata are defined[3] by rules of the form

$$x_i^{t+1} = f(x_{i-1}^t, x_i^t, x_{i+1}^t) \qquad f: \{0, 1\}^3 \to \{0, 1\} \tag{2.1}$$

i.e., the sites can assume either of the values $\{0, 1\}$, and only nearest-neighbor interactions are considered. A rule is therefore equivalently defined by specifying the value assigned to each of the 2^3 possible 3-tuple configurations of site values; i.e., by specifying the a_j, $j = 0,..., 7$, such that

000	001	010	011	100	101	110	111
↓	↓	↓	↓	↓	↓	↓	↓
a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7

(In the remainder of this paper, the assignment of value a_j to a tuple (x, y, z) will often be written as $xyz \to a_j$; e.g., $000 \to 0$ or $011 \to 1$.) Since each $a_j \in \{0, 1\}$, there is a total of $2^{2^3} = 256$ possible elementary rules.

This section will consider the existence of invariant strings for elementary rules. The techniques are applicable in principle, although tedious in practice, to the case of rules with any fixed neighborhood size. Three aspects of the problem will be discussed. First, it will be shown that, given an arbitrary finite string of 0's and 1's, a simple procedure yields the elementary rules for which the string is invariant. Second, the set of rules for which invariant strings exist will be defined, and the set of invariant strings associated with each rule completely specified. Third, the ordering of sets of invariant strings associated with elementary rules will be used to induce an ordering of the rules themselves.

Consider now the question of defining a rule under which a given finite string is invariant.

Notation. * denotes a "wild card" symbol that assumes both of the values $\{0, 1\}$. The use of * in a relation signifies that the relation holds for both $* = 0$ and $* = 1$.

Notation. Let $\{x_0^t x_1^t \cdots x_n^t\}$ denote a string of site values generated by a rule R at some time t. Then the string's image $\{x_1^{t+1} x_2^{t+1} \cdots x_{n-1}^{t+1}\}$ will be written as $R\{x_0^t x_1^t \cdots x_n^t\}$.

Definition. The string $\{x_0^t \cdots x_n^t\} = \{x_0 \cdots x_n\}$ is an *invariant string* of rule R if $R\{*x_0^t \cdots x_n^t *\} = \{x_0^t \cdots x_n^t\} = \{x_0 \cdots x_n\}$.

The following theorems are stated without proof.

Theorem 1. Let $\{(x, y, z)\}$ denote the set of possible 3-tuples of 0's and 1's, and let $\{p_0 \cdots p_n\}$ be a string with $p_i \in \{0, 1\}$ for all $0 \leqslant i \leqslant n$. Define rule R by

$$* \, p_0 p_1 \to p_0, \; p_0 p_1 p_2 \to p_1, \ldots$$

$$p_{n-2} p_{n-1} p_n \to p_{n-1}, \; p_{n-1} p_n * \to p_n \tag{2.2}$$

with arbitrary values assigned to

$$\{a_j = f(x, y, z) \mid (x, y, z) \neq (*, p_0, p_1), (p_{n-1}, p_n, *), (p_{i-1}, p_i, p_{i+1}); 1 \leqslant i \leqslant n\}$$

where f is the function defining the rule in (2.1). Then the string $\{p_0 \cdots p_n\}$ is invariant under R.

Theorem 2. Let R be a rule defined by (2.1). Define

$$G = \{(y, z) \mid f(*, y, z) = y\}$$
$$H = \{(x, y) \mid f(x, y, *) = y\}$$
$$I = \{(x, y, z) \mid f(x, y, z) = y\}$$

Suppose an invariant string exists for R. Then G, H, and I must be nonempty, and there must exist a string $\{x_0 \cdots x_n\}$ where $n > 0$, $(x_0, x_1) \in G$, $(x_{n-1}, x_n) \in H$, and $(x_{i-1}, x_i, x_{i+1}) \in I$ for $1 \leqslant i \leqslant n-1$. Moreover, any such string is an invariant string for R.

Remark 1. There are three essential results in Theorem 2: first, that the only rules with invariant strings are those for which the sets G and H are nonempty; second, that the length of an invariant string must be $\geqslant 2$; and third, that the set of invariant strings for any rule can be deduced from the sets G, H, and I. The third result is obtained by inspection. The first and second results are obtained from the necessary conditions derived in Ref. 4 for the generation of constant temporal sequences by nearest-neighbor rules. Note in particular that it was shown in Ref. 4 that a string of length 1 can in fact be constant under evolution of certain rules, but only under the assumption of particular symmetry properties satisfied by the entire automata. Strings of length 1 therefore do not satisfy the definition used in this paper of invariant strings.

Remark 2. Suppose $P = \{p_0 \cdots p_n\}$ is an invariant string for some rule. There are two possibilities for the generation of P by the automaton: the string P may appear in the initial condition, or it may be generated at some time $t > 0$, and subsequently be preserved by the automaton. The above theorems do not distinguish between the two cases. A mechanism for doing so, in contexts where the objective is to detect the presence of invariant strings in the initial condition, is discussed in Section 3.

Remark 3. If $P = \{p_0 \cdots p_n\}$ is an arbitrary string, then P will be invariant under any rule R defined as in Theorem 1. Clearly, R is not uniquely defined. Of the set of all possible rules R for which P is invariant, the rule that minimizes the number of other invariant strings is the rule that assigns

$$\begin{aligned} f(x, y, z) = y \quad & \text{iff} \quad (x, y, z) \in \{(*, p_0, p_1), (p_{n-1}, p_n, *), \\ & \qquad (p_{i-1}, p_i, p_{i+1}); 1 \leqslant i \leqslant n\} \\ = y' \quad & \text{otherwise} \end{aligned} \tag{2.3}$$

where $y' \neq y$. The following definition is therefore motivated.

Definition. Let $P = \{p_0 \cdots p_n\}$ be an arbitrary string. Then the *elementary rule induced by P* is the rule uniquely defined by (2.2) and (2.3)

Theorems 1 and 2, and the remarks that follow them, establish that, given an arbitrary string of values, it is indeed possible to construct the rules of radius 1 such that the string, together with a well-defined set of related strings, is invariant under the rules. (In fact, as will be discussed in Section 4, it is possible to do so for any specified radius.) Moreover, there is a unique rule (the rule induced by the string) that minimizes the number of other invariant strings. From Theorem 2, it can be seen that the class of elementary rules for which invariant strings exist can be specified exactly. A straightforward calculation using the Inclusion-Exclusion Principle shows that the number of elementary rules with invariant strings is 83.

It is useful at this point to introduce the concept of a "string-generating" graph, examples of which are shown in Fig. 1.[2] The possible nodes of the graph are restricted to the set $\{00, 01, 10, 11\}$. Consider now any elementary rule R and the set I defined in Theorem 2. For each tuple $(x, y, z) \in I$, the nodes xy and yz are included in the graph, and an edge labeled z is drawn from node xy to node yz. Note that any elementary automaton rule (not merely those with invariant strings) can be assigned in this fashion a unique graph. Thus, for example, the "identity" rule (rule number 204 in the labeling scheme of Ref. 3) for which $f(x, y, z) = y$ for all possible values of x, y, and z, is represented as shown in Fig. 1a, and its graph is the maximal possible graph. At the other extreme, the rule assigning $\{000, 001, 100, 101\} \to 1$, $\{010, 011, 110, 111\} \to 0$, is represented by the null graph.

The advantage of string-generating graphs is that they facilitate construction of the set of invariant strings for any rule. In general, a string (not necessarily invariant) is generated by traversing a path in the graph. The label of the "start" node of the path provides the first two values in the string, and the label of each subsequent edge used by the path is then appended to the right of the string. Then Theorem 2 states that rule R has an invariant string iff there exists in the graph of R (i) a node $(xy)'$ with incoming degree 2; (ii) a node $(xy)''$ with outgoing degree 2; (iii) a path between $(xy)'$ and $(xy)''$.

Moreover, any path beginning at $(xy)'$ and ending at $(xy)''$ represents an invariant string of the rule. Clearly, if the graph is such that (a) there exists a node xy with both incoming and outgoing degree 2, (b) every other node has both incoming and outgoing degree < 2, and (c) there is only one circuit beginning and ending at node xy, then every invariant string of the

[2] I thank Stephen Wolfram for pointing out that graphs of this type have previously been studied by N. G. de Bruijn and I. J. Good.

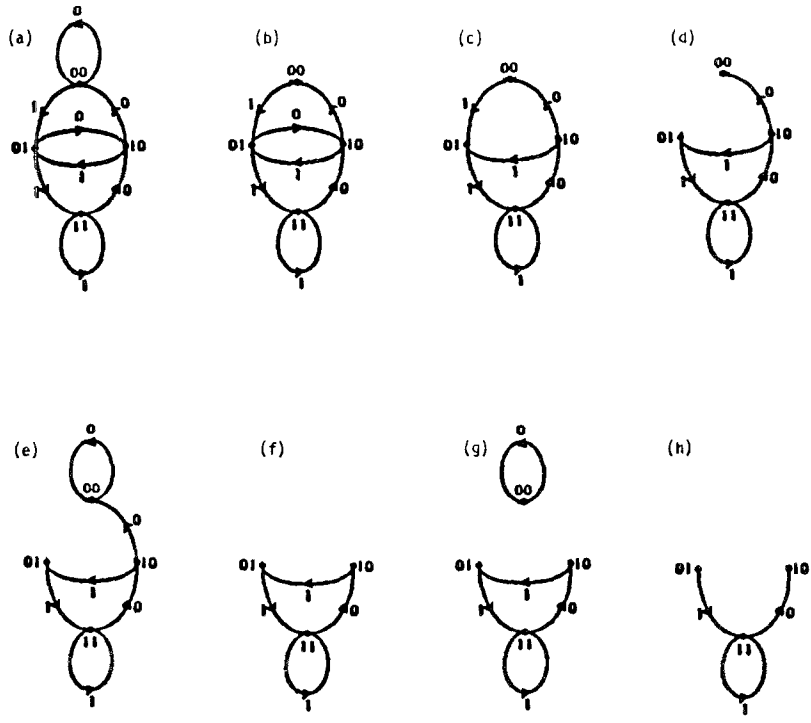

Fig. 1. String-generating graphs. For each tuple xyz for which a rule R assigns $f(x, y, z) = y$, the pairs xy and yz are included as nodes in the graph, and an edge labeled z is drawn from xy to yz. An invariant string for R is generated by choosing any node with incoming degree 2, continuing along any possible path for an arbitrary number of steps, and eventually terminating at any node with outgoing degree 2. The label of the "start" node is taken to be the first two values of the invariant string; each subsequent edge label for the path chosen is then appended on the right. (a) Rule 204: $\{000, 001, 100, 101\} \to 0$, $\{010, 011, 110, 111\} \to 1$; (b) Rule 205: $\{001, 100, 101\} \to 0$, $\{000, 010, 011, 110, 111\} \to 1$; (c) Rule 201: $\{001, 010, 100, 101\} \to 0$, $\{000, 011, 110, 111\} \to 1$; (d) Rule 202: $\{010, 100, 101\} \to 0$, $\{000, 001, 011, 110, 111\} \to 1$; (e) Rule 203: $\{000, 010, 100, 101\} \to 0$, $\{001, 011, 110, 111\} \to 1$; (f) Rule 218: $\{010, 101\} \to 0$, $\{000, 001, 011, 100, 110, 111\} \to 1$; (g) Rule 219: $\{000, 010, 101\} \to 0$, $\{001, 011, 100, 110, 111\} \to 1$; (h) Rule 251: $\{010\} \to 0$, $\{000, 001, 011, 100, 101, 110, 111\} \to 1$.

rule will be a repeated concatenation of the invariant string generated by traversing the circuit exactly once (see Fig. 1h). If, however, there exists more than one circuit beginning and ending at node xy, then the invariant strings can assume a more varied form (see Fig. 1a–g). Furthermore, given a graph for a rule R satisfying conditions (i)–(iii), it is clear that a rule R' will share a common set of invariant strings with R iff the graph of R' does

not include any nodes or edges that permit a path different from those obtainable from the graph of R. It is easy to see, for example, by comparing the graph of Fig. 1a (representing the complete graph) and that of Fig. 1h that there are 12 distinct rules for which strings of 1's constitute the only possible invariant strings.

The set of invariant strings for an elementary automaton can equivalently be characterized by specifying blocks of values that may not occur in the strings, together with the permissible beginning and ending string values. (See Ref. 5 for a discussion of the relation between excluded blocks and invariant strings.) For example, the sets of invariant strings shown in Fig. 1 can alternatively be described as

(a) all strings

(b) strings beginning and ending with 01, 10, or 11; and containing no 000

(c) strings beginning with 01 or 11, ending with 10 or 11; and containing neither 000 nor 010

(d) strings beginning with 11, ending with 10 or 11; and containing neither 010 nor 00

(e) same as d

(f) strings beginning with 11, ending with 11; and containing neither 010 nor 00

(g) same as f

(h) all strings containing no 0

Table I provides a list of all 83 elementary rules for which invariant strings exist, grouped according to their excluded blocks.

A hierarchy of rules for which invariant strings exist is shown in Fig. 2. In the graph, each node N represents the set C of rules in whose invariant strings the block N of values may not occur. (Refer to Table I for a list of rules belonging to each set.) An edge is drawn between sets C_i and C_j if any string that is an invariant string for a rule belonging to set C_j is also an invariant string for a rule belonging to C_i. Each path of the originating from the node "NONE" (representing the identity rule) corresponds to a particular choice of $x, y, z \in \{0, 1\}$ for which the conditions of Theorem 2 are satisfied. The ordering along each path reflects an ordering of the number of distinct permissible paths in the string-generating graphs of the corresponding rules; i.e., an ordering of the number of tuples (x, y, z) for which the rule assigns $f(x, y, z) = y$.

Table I. List of Elementary Rules for which Invariant Strings Exist[a]

Excluded blocks	Rule numbers	Beginning values	Ending values
none	204	0, 1	0, 1
000	205	01, 10, 11	01, 10, 11
010	200	00, 01, 11	00, 10, 11
101	236	00, 10, 11	00, 01, 11
111	76	00, 01, 10	00, 01, 10
000, 010	201	01, 11	10, 11
000, 101	237	10, 11	01, 11
000, 111	77	01, 10	01, 10
010, 101	232	00, 11	00, 11
010, 111	72	00, 01	00, 10
101, 111	108	00, 10	00, 01
000, 010, 101	233	11	11
000, 010, 111	73	01	10
000, 101, 111	109	10	01
010, 101, 111	104	00	00
000, 11 (a)	13, 141	01, 10	10
(b)	19, 69	01	01, 10
(c)	5, 133	01	10
111, 00 (a)	78, 79	10	01, 10
(b)	92, 93	01, 10	01
(c)	94, 95	10	01
010, 00 (a)	202, 203	11	10, 11
(b)	216, 217	01, 11	11
(c)	218, 219	11	11
101, 11 (a)	100, 228	00	00, 01
(b)	44, 172	00, 10	00
(c)	36, 164	00	00
00, 11 (a)	70, 71, 198, 199	10	10
(b)	28, 29, 156, 157	01	01
00 (a)	206, 207	10, 11	01, 10, 11
(b)	220, 221	01, 10, 11	01, 11
(c)	222, 223	10, 11	10, 11
11 (a)	68, 196	00, 01, 10	00, 10
(b)	4, 28	00, 01	00, 01, 10
(c)	132, 140	00, 01	00, 10
0	234, 235, 238, 239, 248, 249 250, 251, 252, 253, 254, 255	1	1
1	0, 8, 32, 40, 64, 96, 128 136, 160, 168, 192, 224	0	0

[a] Grouped according to the values that may not occur in the invariant strings and their permissible beginning and ending values. For example, any string that begins and ends with 10 and does not contain either 00 or 11, will be an invariant string of rules 70, 71, 198, and 199.

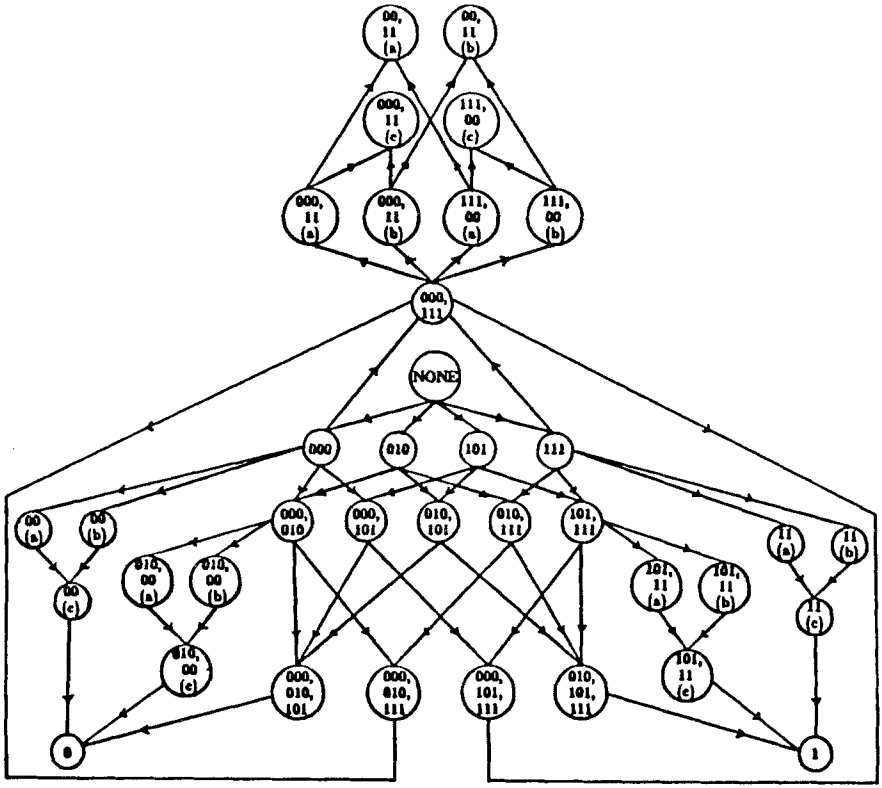

Fig. 2. Hierarchy of rules for which invariant strings exist. Each node N represents a set C of rules whose invariant strings exclude block N of values. (See Table I for a list of rules belonging to each set.) An edge is drawn between sets C_i and C_j of rules if any string that is invariant for a rule in set C_j is also invariant for a rule in C_i.

3. FILTERING RULES AND APPLICATIONS TO PATTERN RECOGNITION

This section will discuss the extension of elementary cellular automata rules to "filtering" rules. Filtering rules can literally be regarded as a subclass of rules that depend on nearest-neighbor interactions, but that are defined on sites assuming any of three values rather than the usual two. The third value can conveniently be interpreted as a "blank." The rules are termed "filtering" since they possess the property that arbitrary initial conditions evolve to (or are "filtered" for) spatial sequences composed of concatenations of the invariant strings for elementary rules, separated by blanks. The blanks themselves are invariant under the filtering rule, and the

spatial sequence as a whole is therefore a fixed point for the automaton. Moreover, the invariant strings that appear in the spatial sequence are precisely the invariant strings present in the automaton's initial condition. Thus, given any initial condition for a filtering rule, it is possible to specify immediately the fixed point to which it will be attracted. The applications of filtering rules to pattern recognition will be discussed.

Definition. The *extended set* V^- of values that can be assumed by the sites of a cellular automaton (1.1) is defined as $V^- = V \cup \{\_\}$, where $V = \{0,..., k-1\}$ and $\_$ denotes "blank."

Definition. Let $\{x_i^0\}$ be an arbitrary finite initial condition such that $x_i^0 = 0$ for $i < M$, $i > N$, and $x_M^0 = x_N^0 = 1$. Then the *truncated initial condition* is defined as $\{y_i^0\}$ with

$$\begin{aligned} y_i^0 &= x_i^0 \qquad M \leqslant i \leqslant N \\ &= \_ \qquad \text{otherwise} \end{aligned}$$

Definition. Let R be an elementary cellular automaton rule defined by (2.1). Then define the *filtering rule* R^- to be

$$x_i^{t+1} = f^-(x_{i-1}^t, x_i^t, x_{i+1}^t) \qquad f\colon \{\_, 0, 1\}^3 \to \{\_, 0, 1\} \tag{3.1}$$

with

$$\begin{aligned} f^-(x, y, z) &= y \qquad \text{if} \quad f(x, y, z) = y & (3.2)\\ &= \_ \qquad \text{if} \quad f(x, y, z) \neq y \\ f^-(\_, y, z) &= y \qquad \text{if} \quad f(*, y, z) = y & (3.3)\\ f^-(x, y, \_) &= y \qquad \text{if} \quad f(x, y, *) = y \\ f^-(x, y, z) &= \_ \qquad \text{otherwise} & (3.4) \end{aligned}$$

Theorem 3. Let S be the set of invariant strings for the elementary rule R. Assume the initial condition has the form

$$\{x_i^0, M \leqslant i \leqslant N\} = \{D_0 B_1 D_1 \cdots D_{j-1} B_j D_j \cdots B_n D_n\}$$

where $B_j \in S$, $D_j \notin S$ for all j and $n \geqslant 0$. Then for $t \geqslant \max \text{len}(D_j)$, the spatial sequences obtained by applying the filtering rule R^- to the truncated initial condition are of the form

$$\{\_\_\_\_B_1\_B_2\_\cdots\_B_n\_\_\_\_\}$$

Proof. Consider the images under the filtering rule R^- of the string D_j. For $j=1,\ldots,n-1$, the string $D_j=\{d_1 d_2 \cdots d_{q-1} d_q\}$, $q=\text{len}(D_j)$, is "embedded" in the configuration $B_j D_j B_{j+1}$. Let $B_j=\{b_1 b_2 \cdots b_{p-1} b_p\}$ and $B_{j+1}=\{c_1 c_2 \cdots c_{r-1} c_r\}$ for some p, r, and define $d_0=b_p$, $d_{q+1}=c_1$. Then Theorem 2 implies that

$$f(d_{k-1}, d_k, d_{k+1}) \neq d_k$$

for some $1 \leqslant k \leqslant q$, since otherwise the string would be invariant. Furthermore, by the same reasoning, for any $1<m<q$ there exists a $y \in \{0, 1\}$ such that

$$f(d_{m-2}, d_{m-1}, y) \neq d_{m-1}$$

and a $z \in \{0, 1\}$ such that

$$f(z, d_{m+1}, d_{m+2}) \neq d_{m+1}$$

Hence the filtering rule R^- assigns

$$f^-(d_{k-1}, d_k, d_{k+1}) = -$$

and furthermore.

$$f^-(d_{k-2}, d_{k-1}, -) = -$$

i.e., the blanks propagate to the left, and

$$f^-(-, d_{k+1}, d_{k+2}) = -$$

i.e., the blanks propagate to the right. Thus in a finite number of time steps bounded above by $\text{len}(D_j)$, the string D_j will be mapped onto a string of blanks. The argument for D_0 and D_n is similar. For these cases, the use of the truncated initial condition implies that the strings are embedded as shown

$$--D_0 B_1 \qquad \text{and} \qquad B_n D_n--$$

The proof then proceeds as for the first case.

Remark. It is mentioned in Section 2 that the invariant strings generated by automata rules are of two types: those that are present in the initial condition at time $t=0$, and those that are generated at some later

time $t>0$. Theorem 3 establishes that for "filtering" rules, an invariant string of the associated elementary rule will be preserved only if it appears in the initial condition. All other strings (in particular, those that would, under evolution of the elementary rule, generate invariant strings at some later time) will be "annihilated." Thus, every initial condition for a filtering rule will evolve to a spatial sequence composed of invariant strings appearing precisely where they did in the initial condition, with blanks everywhere else. Since the blanks themselves are invariant, the spatial sequence is a fixed point for the rule, and spatial sequences of this form are the only attractors of the automaton.

In the special case that an elementary rule has been induced by a particular string, the definition of the associated filtering rule can be modified slightly so as to restrict even further the spatial sequences that serve as attractors for the rule.

Definition. Let R be the elementary rule induced by the string $\{p_0 \cdots p_n\}$. Define the *string-filtering rule* $R^-(p_0 p_1, p_{n-1} p_n)$ to be the rule satisfying conditions (3.1), (3.2), and (3.4), but with condition (3.3) replaced by

$$\begin{aligned} f^-(-, p_0, p_1) &= p_0 \\ f^-(p_{n-1}, p_n, -) &= p_n \end{aligned} \tag{3.3a}$$

Theorem 4. Let R be the elementary rule induced by the string $\{p_0 \cdots p_n\}$, and let $S(p_0 p_1, p_{n-1} p_n)$ be the set of invariant strings of R that begin with the values $p_0 p_1$ and end with the values $p_{n-1} p_n$. Then the spatial sequences obtained by applying the string-filtering rule $R^-(p_0 p_1, p_{n-1} p_n)$ to the truncated initial condition consist of strings belonging to $S(p_0 p_1, p_{n-1} p_n)$ separated by blanks.

Proof. The proof follows along the lines as for Theorem 3.

Consider now the use of cellular automata in pattern recognition. The relevant problem is the "recognition" by an automaton of a specific pattern, or string of 0's and 1's, in an arbitrary input sequence. Denote the pattern by $\{p_0 \cdots p_n\}$. Theorem 1 of this paper states that the pattern will be "recognized," or preserved, by any rule R defined with ${}^*p_0 p_1 \to p_0, \ldots, p_{n-1} p_n {}^* \to p_n$, and the other values a_j to be assigned by the rule left unconstrained. Theorem 2 then provides the set of other strings that will also be recognized by the rule R. The rule induced by the string is defined as the unique rule that assigns $a_j = f(x, y, z) \neq y$ for $\{xyz\}$ not appearing in $\{p_0 p_1 \cdots p_{n-1} p_n\}$, thus minimizing the number of other patterns preserved. From Theorem 4, the associated string-filtering rule

possesses the property that only those invariant strings that begin and end with the same values as the pattern, and that appear in the initial condition, will be preserved by the automaton. Any other string, representing "noise" in the input, will be annihilated. Note that the computation of site values can be performed in parallel.

Figure 3 provides an example illustrating the use of filtering rules. The pattern 10011 is to be recognized, and the input sequence is taken to be a random sequence of 0's and 1's containing that pattern, along with others. In accordance with Theorems 2 and 4, the rule is chosen to be

$$\{001, 100\} \to 0; \qquad \{010, 011, 110, 111\} \to 1; \qquad \{\_10, 11\_\} \to 1$$

with all other tuples assigned the value _. The associated elementary rule number is therefore 237 (according to the labeling scheme of Ref. 3). The string-generating graph of the rule is given by Fig. 3c, and from Table I, it can be seen that any string beginning with 10, ending with 11, and not containing either 000 or 101, will be preserved by the rule.

```
pattern to be recognized:

10011

radius of rule used:   1

input sequence:

01110011011100100110001100111111001101000010100011101010011011111110101000


evolution of automaton:

01110011011100100110001100111111001101000010100011101010011011111110101000
 1110011 11100100110 01100111110011 10  01 10 0111 1 10011 111111 1 10
  110011  110010011   1100111110011 1      1   111   10011  11111   1
   10011   10010011    100111110011            11   10011   1111
   10011   10010011    100111110011             1   10011    111
   10011   10010011    100111110011                 10011     11
   10011   10010011    100111110011                 10011      1
   10011   10010011    100111110011                 10011
   10011   10010011    100111110011                 10011
   10011   10010011    100111110011                 10011
   10011   10010011    100111110011                 10011
```

Fig. 3. Example of filtering. The pattern to be recognized is the string 10011 embedded in an arbitrary input sequence. An appropriate elementary rule is extended to provide an automaton that generates blanks as well as 0's and 1's. An automated procedure constructs both the rule and its extension so as to guarantee that the string will be preserved, and the number of other preserved patterns minimized. Under evolution of the extended rule, the desired pattern, along with any string beginning with 10, ending with 11, and not containing any occurrences of either 000 or 101, is preserved, and all others annihilated in a small number of time steps.

4. MINIMAL-RADIUS FILTERING RULES

The filtering rules described in Section 3 were based on elementary cellular automata involving only nearest-neighbor interactions. The attractors for these rules were shown to be spatial sequences composed of concatenations (separated by blanks) of the invariant strings for the associated elementary rules. Since, as was shown in Section 2, an elementary rule may possess multiple distinct invariant strings, the attractors are therefore highly nonunique. In this section, it will be shown that given an arbitrary string, it is possible to define a rule of minimal neighborhood size for which the string will be the unique invariant string, and therefore every initial condition evolves to a spatial sequence consisting only of occurrences of that string together with blanks. Moreover, an upper bound (depending only on the invariant string itself) is provided for the number of iterations required to attain the final state.

The following theorem is an obvious generalization of Theorem 1.

Theorem 5. Let R be a rule of radius r defined by

$$x_i^{t+1} = f(x_{i-r}^t, \ldots, x_i^t, \ldots, x_{i+r}^t)$$

and let $\{p_0 \cdots p_n\}$ be an arbitrary string. Define the set J to be the set of tuples $(q_{-r}, \ldots, q_r)$ obtained by setting, for each fixed $k = 0, \ldots, n$

$$\begin{aligned} q_j &= p_{k+j} \qquad j = -r, \ldots, 0, \ldots, r \qquad \text{if} \quad 0 \leqslant k+j \leqslant n \\ &= * \qquad\quad \text{otherwise} \end{aligned}$$

Suppose the function f assigns

$$q_{-r} \cdots q_0 \cdots q_r \to q_0$$

for each tuple in J. Then the string $\{p_0 \cdots p_n\}$ is an invariant string for R.

As is the case for nearest-neighbor rules, the rules defined in Theorem 5 are not unique, and must be further constrained in order to obtain the rule that minimizes the number of other invariant strings.

Definition. The *rule of radius r induced by the string* $\{p_0 \cdots p_n\}$ is the unique rule defined as in Theorem 5 that assigns, in addition

$$x_{i-r} \cdots x_i \cdots x_{i+r} \to x_i'$$

where $x_i' \neq x_i$, for all tuples $(x_{i-r}, \ldots, x_{i+r}) \notin J$.

The induced rule of radius r can be extended in the obvious way to define an associated string-filtering rule R^-. The objective of this section is then to derive the minimum value of the radius r for which the given string will constitute the unique invariant string of the induced rule of radius r. In this context, "uniqueness" of an invariant string P is to be understood as implying that the only strings invariant under the rule consist of concatenations (possibly overlapping) of P. In order to eliminate ambiguity in the definition of a rule, it will be assumed throughout that the neighborhood of the rule is of odd length; i.e., the length of the rule is given by $2r+1$.

The next theorem provides an obvious upper bound on the size of the neighborhood needed to guarantee uniqueness of the invariant string.

Theorem 6. The string $\{p_0 \cdots p_n\}$ is the unique invariant string of the induced rule of neighborhood radius $r \geqslant n$.

The results that follow will provide the exact value of the minimal neighborhood size needed to guarantee uniqueness of the invariant string. The major tool used in the derivation of these results will be the characterization of an arbitrary string by the length of its longest self-match, defined to be the length of the longest tuple that occurs more than once in the string.

Definition. A string $\{p_0 \cdots p_n\}$ possesses a *self-match* of length $m > 0$ if there appear in the string two tuples $(p_i, \ldots, p_{i+m-1})$ and $(p_j, \ldots, p_{j+m-1})$ for which $p_{i+k} = p_{j+k}$, $i \neq j$, $k = 0, \ldots, m-1$.
Now define

$m_1 =$ length of longest self-match with either $i = 0$ and $j + m - 1 < n$

or $i > 0$ and $j + m - 1 = n$

$m_2 =$ length of longest self-match otherwise

(i.e., either $i = 0$ and $j + m - 1 = n$, or $i > 0$, and $j + m - 1 < n$)

Set

$$\begin{aligned} r_1 &= m_1 + 1 \\ r_2 &= (m_2 + 1)/2, \qquad m_2 \text{ odd} \\ &= (m_2 + 2)/2, \qquad m_2 \text{ even} \end{aligned} \tag{4.1}$$

Then the following theorem provides the desired result.

Theorem 7. Let $P = \{p_0 \cdots p_n\}$ be a string. Then the induced rule of radius $r^* = \max(r_1, r_2)$, with r_1 and r_2 defined as in (4.1), is the minimal-radius rule for which the string P is the unique invariant string.

Proof. First show that an induced rule R of radius $r < r^*$ will have more than one distinct invariant string. Suppose that $r^* = r_1$, and assume that $m = r_1 - 1$ is the largest value such that there exists a tuple $(p_0,..., p_{m-1})$ that "matches" the tuple $(p_j,..., p_{j+m-1})$ for some $0 < j < n - m + 1$. The case being considered is therefore $i = 0$ and $j + m - 1 < n$. The strategy for this case is to show that $X = \{x_k\}$, $k = 0,..., n + j - 1$, where

$$\begin{aligned} x_k &= p_k & k &= 0,..., j + m - 1 \\ &= p_{k-j} & k &= j + m,..., n + j - 1 \end{aligned}$$

is an invariant string of R different from $P = \{p_0 \cdots p_n\}$. Note that the string X is constructed so that its first $j + m$ values coincide with those of P, but the values to the right of p_{j+m-1} in string X are those that follow p_{m-1} in string P; i.e., the block $\{p_m \cdots p_{j+m-1}\}$ appears twice in X. Since $(p_0 \cdots p_{m-1})$ matches $(p_j \cdots p_{j+m-1})$, every subtuple of length $2m + 1$ in X appears also in P; that is, locally (up to length $2m + 1$), the string X appears identical to P. Hence if the rule has radius $r \leqslant m$, the invariant string P will not be unique. The case where $r^* = r_1$, $i > 0$, and $j + m - 1 = n$ is the same.

Now suppose that $r^* = r_2$, and assume that m is the largest value such that there exist a match between two tuples $(p_i,..., p_{i+m-1})$ and $(p_j,..., p_{j+m-1})$ for $i > 0$ and $j + m - 1 < n$. Show that $X = \{x_k\}$, $k = 0,..., n + i - j$, where

$$\begin{aligned} x_k &= p_k & k &= 0,..., i + m - 1 \\ &= p_{k-i+j} & k &= i + m,..., n + i - j \end{aligned}$$

is an invariant string of R. String X is constructed so that its first $i + m$ values coincide with those of P, but the values to the right of p_{i+m-1} are those that follow p_{j+m-1} in P; i.e., the block $p_{i+m} \cdots p_{j+m-1}$ does not appear in X. Then every subtuple of length $m + 1$ in X appears somewhere in P. Hence, if the rule has radius $r \leqslant m/2$, the invariant string P will not be unique.

The case in which $i = 1$ and $j + m - 1 = n$ can be regarded as a special case of the previous one where now the string x is taken to be $\{x_k\}$, $k = 0,..., m - 1$, with

$$x_k = p_k \qquad k = 0,..., m - 1$$

It has thus been shown that any induced rule R of radius $r < r^*$ will have an invariant string different from P. Next prove that the rule R of radius $r = r^*$ has string P as its unique invariant string.

Define the set I to be

$$\{(x_{-r}\cdots x_0\cdots x_r)\,|\,f(x_{-r},\ldots,x_0,\ldots,x_r)=x_0\}$$

where f is the function defining the rule. From the construction of the induced rule, it is clear that set I consists of

$$\begin{array}{ll} 2^r \text{ tuples} & (*\cdots * p_0\cdots p_r) \\ 2^{r-1} \text{ tuples} & (*\cdots * p_0\cdots p_{r+1}) \\ \quad\vdots & \\ 1 \text{ tuple} & (p_0\cdots p_r\cdots p_{2r}) \\ \quad\vdots & \\ 2^{r-1} \text{ tuples} & (p_{n-r-1}\cdots p_n *\cdots *) \\ 2^r \text{ tuples} & (p_{n-r}\cdots p_n *\cdots *) \end{array} \tag{4.2}$$

where, as always, * denotes both of the values 0 and 1. (Note that a given tuple may appear more than once in the above list.) Suppose that $r=r^*=r_2$, and let $\{q_0q_1\cdots\}$ represent an invariant string of the rule. It will first be shown that $q_k=p_k$ for $k=0,\ldots,r$.

Assume the contrary. Then from the definition of invariant strings, it must be true that

$$*\cdots * q_0\cdots q_r \to q_0 \tag{4.3}$$

i.e., all 2^r tuples of length $2r+1$ with $(q_0\cdots q_r)$ as their final $(r+1)$ values must belong to the set I. The strategy of the proof is to show that the assumption

$$(* p' p'\cdots p' q_0\cdots q_r)\in I \tag{4.4}$$

where $p'\neq p_0$, leads to a contradiction. The first step is to show that the two tuples $\{* p'\cdots p' q_0\cdots q_r\}$ of length $2r+1$ must both appear in string P. Suppose $\{q_0\cdots q_r\}$ matches $\{p_i\cdots p_{i+r}\}$ for some $0<i\leqslant r-1$. Since $p'\neq p_0$, relation (4.4) cannot be satisfied using any of the tuples in (4.2) with the "wild card" symbol * on the left. Next suppose $\{q_0\cdots q_s\}$ matches $\{p_{n-s}\cdots p_n\}$ for some $0\leqslant s<r$. If (4.4) is to be true, then $p_{n-s-h}=p'$ for $h=1,\ldots,r$. Furthermore, suppose $p_{n-s-r}=c$. Then a tuple with its first $(r+1)$ elements equal to $(c' p'\cdots p' q_0)$ must belong to set I. This would imply that $m_1\geqslant r+1$, and hence $r^*\neq r_2$, a contradiction. The conclusion is that relation (4.4) cannot be satisfied using any of the tuples in (4.2) with the "wild card" symbol * on the right, and hence the tuples $\{* p'\cdots p' q_0\cdots q_r\}$ must appear in the string P for $q_0=p_k$, $r\leqslant k\leqslant n-r$. The length of the longest self-match must therefore be $\geqslant 2r$. The contradiction shows that $\{q_0\cdots q_r\}$ matches $\{p_0\cdots p_r\}$.

Finally show that $q_{r+1} = p_{r+1}$. Suppose that $q_{r+1} \neq p_{r+1}$. Then

$$(* \cdots * q_0 q_1 \cdots q_r q_{r+1})$$

and

$$(* \cdots * q_0 q_1 \cdots q_r q'_{r+1})$$

where $q'_{r+1} \neq q_{r+1}$, both belong to the set I. But the $(r+1)$-tuple $(q_0 \cdots q_r)$ cannot occur twice in P, since otherwise r^* cannot equal r_2. Hence $q_{r+1} = p_{r+1}$.

By induction, the string $\{p_0 \cdots p_n\}$ is the only invariant string of the rule. The case for $r^* = r_1$ is similar.

Corollary 7. Let R^- be the filtering rule of radius r^* for which P is the unique invariant string. Then the number of iterations required for an arbitrary initial condition to evolve to its limiting state is bounded above by

$$\left[\frac{\ln(P) - 1}{r^*}\right]'$$

where $[x]'$ denotes the smallest integer greater than or equal to x.

Remark. In the context of pattern recognition, Theorem 7 provides the neighborhood size of the optimally efficient filtering rule that will

Table II. Examples of Minimal Radius Size Needed to Filter for Given Patterns[a]

Pattern	Minimal radius
111111	3
011111	5
000111	3
010101	3
001110	2
001111	4
1101011000110000	4
String of N 1's (N even)	$N/2$
String of M 0's followed by N 1's ($M < N$, $M + N$ even)	N
Periodic string length N, period p (N, p even)	$1 + (N - p)/2$

[a] The radius sizes shown are calculated using Theorem 7. Using these radii, filtering rules can be constructed such that the desired pattern is preserved, and all others annihilated. For any rule with radius less than that shown, the preserved patterns will not be unique.

```
pattern to be recognized:   011111

input sequence:
01110101101111111010111001110001110111111010110111111100000100110101010110000

radius of rule used:   2

evolution of automaton:

01110101101111111010111001110001110111111010110111111100000100110101010110000
01 1  0  0111111  01 1 01 1  01 1011111  0  0111111       0      0
         0111111                 011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111

radius of rule used:   3

evolution of automaton:

01110101101111111010111001110001110111111010110111111100000100110101010110000
0        0111111  0    0     0   011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111

radius of rule used:   4

evolution of automaton:

01110101101111111010111001110001110111111010110111111100000100110101010110000
         0111111                 011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111
         0111111                 011111     0111111

radius of rule used:   5

evolution of automaton:

01110101101111111010111001110001110111111010110111111100000100110101010110000
         011111                  011111     011111
         011111                  011111     011111
         011111                  011111     011111
         011111                  011111     011111
         011111                  011111     011111
         011111                  011111     011111
```

(a)

Fig. 4. Examples of filtering for a given pattern using rules of varying neighborhood size. Each rule is uniquely constructed so as to guarantee that the pattern, together with the minimal possible number of other patterns, will be invariant under the rule. Theorem 7 provides the minimal radius of the rule for which the pattern will be preserved and all strings annihilated. (a) The pattern to be recognized is the 16-bit long string 1101011000110000 embedded in an arbitrary input sequence. Rules of radius varying from 1 to 4 are used to filter for the pattern. Theorem 7 states that the rule of radius 4 is the minimal-radius rule that annihilates all strings other than the desired pattern. (b) The pattern to be recognized is the six-bit long string 011111. Theorem 7 states that the minimal-radius rule has radius 5.

```
pattern to be recognized:   1101011000110000

input sequence:
01110101101101011000110000111011111010110000001001101010110000101011100000


radius of rule used:   1

evolution of automaton:

01110101101101011000110000111011111010110000001001101010110000101011100000
 11101011011010110001100001110111101011000000100110101011000010101110000
 11101011011010110001100001110111101011000000100110101011000010101110000
 11101011011010110001100001110111101011000000100110101011000010101110000
 11101011011010110001100001110111101011000000100110101011000010101110000
 11101011011010110001100001110111101011000000100110101011000010101110000
 11101011011010110001100001110111101011000000100110101011000010101110000
 11101011011010110001100001110111101011000000100110101011000010101110000


radius of rule used:   2

evolution of automaton:

01110101101101011000110000111011111010110000001001101010110000101011100000
  1101011 11010110001100000 11   1101011000000   1101 10110000  10 1 0000
  11010   1101011000110000       1101011000000   11     110000          00
  110     1101011000110000       1101011000000           1 0000
  1       1101011000110000       1101011000000              00
          1101011000110000       1101011000000
          1101011000110000       1101011000000


radius of rule used:   3

evolution of automaton:

01110101101101011000110000111011111010110000001001101010110000101011100000
  110101  11010110001100000 1    1101011000000   110   0110000         000
  110     1101011000110000       1101011000000             0000
          1101011000110000       1101011000000                0
          1101011000110000       1101011000000
          1101011000110000       1101011000000
          1101011000110000       1101011000000


radius of rule used:   4

evolution of automaton:

01110101101101011000110000111011111010110000001001101010110000101011100000
  1101    1101011000110000       110101  000     11     1  000          00
          1101011000110000       11
          1101011000110000
          1101011000110000
          1101011000110000
          1101011000110000
```

(b)

Fig. 4 (*continued*)

preserve all occurrences of a desired pattern, and annihilate all others. In other words, once the minimal neighborhood size has been computed, the induced rule of that neighborhood size can be easily constructed, and then extended to provide a string-filtering rule of the same neighborhood size. This string-filtering rule has the property that within a finite number of iterations bounded above by the result in Corollary 7, arbitrary initial conditions are mapped to spatial sequences consisting of occurrences of the desired pattern, separated by blanks. Any string-filtering rule of a smaller neighborhood size will preserve patterns other than the desired pattern.

Table II contains some examples of the size of the rule necessary and sufficient to guarantee that a particular string will be the unique invariant string. Figure 4a illustrates the recognition of the pattern $\{1101011000110000\}$ using string-filtering rules with varying radii; Fig. 4b illustrates the same process for the pattern $\{011111\}$.

5. SUMMARY

An invariant string of an automaton rule R is defined to be a finite spatial sequence of adjacent site values that remains invariant under the evolution of R, independent of the sequence's spatial position or the values of its neighboring sites. This paper has considered two aspects of invariant strings; namely, the existence of invariant strings for automata rules with a fixed neighborhood size (in particular, elementary, or nearest-neighbor, rules), and the identification of automata rules of unspecified neighborhood size for which an arbitrarily chosen string is invariant. With respect to the first question, it has been shown that there are 83 elementary rules for which invariant strings exist, and a complete specification of the set of invariant strings given for each such rule. The ordering of the sets of invariant strings serves as a basis for inducing an ordering of elementary rules possessing invariant strings.

Automata rules for which invariant strings exist can be extended to define "filtering" rules defined with the same neighborhood size, but with the set of possible site values extended to include a "blank." Arbitrary finite initial conditions are then mapped in finite time to spatial sequences consisting of invariant strings for the original automata, separated by blanks. The invariant strings in the limiting spatial sequence appear precisely where they did in the initial condition. Thus concatenations of invariant strings together with blanks constitute, for these rules, the only attractors of the automata.

Finally, the concept of the "longest self-match" in an arbitrary string has been introduced. The longest self-match is defined to be the longest tuple that occurs more than once in the string. It has been shown that the

length of the longest self-match characterizes the minimal neighborhood size of the rule for which the given string will represent the unique invariant string. Thus, for the filtering rule obtained by extending this "minimal-radius" rule, every initial condition evolves to a spatial sequence composed of occurrences of the given string, separated by blanks. Moreover, the automaton attains its limiting state in a finite number of steps bounded above by a quantity depending only on the invariant string itself.

The results of this paper can be viewed as describing certain pattern-recognizing capabilities of one-dimensional cellular automata. Given a specific pattern embedded in an arbitrary input sequence, a rule of any given neighborhood size can be chosen that preserves that pattern, along with a well-defined set of related patterns, and annihilates all others. Furthermore, a rule of minimal neighborhood size, representing an optimally efficient filter, can be chosen to ensure that the desired pattern will be the only pattern preserved by the automaton.

It is noteworthy that the study of invariant strings represents a problem that relies heavily, in its formulation and solution, on mathematical features of discreteness and local interaction peculiar to cellular automata. For example, the specification of all automata for which a given string is invariant represents a problem that would not be well-posed in the context of, say, differential equations. The results on invariant strings thus provide further evidence of the mathematical richness as well as the computational applicability of cellular automata theory.

ACKNOWLEDGMENTS

I thank Stephen Wolfram and George Zweig for useful discussions.

REFERENCES

1. J. von Neumann, *Theory of Self-reproducing Automata*, A. W. Burks, ed. (Univ. Illinois Press, Urbana, 1966).
2. D. Farmer, T. Toffoli, and S. Wolfram, eds, "Cellular Automata: Proceedings of an Interdisciplinary Workshop," *Physica* **10D** (1–2) (1984).
3. S. Wolfram, *Physica* **10D**:1 (1984).
4. E. Jen, "Global properties of cellular automata," *J. Stat. Phys.* **43**:219 (1980).
5. S. Wolfram, *Comm. Math. Phys.* **96**:15 (1984).