MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence
Memo. No. 166 September 1968

Recognition of Topological Invariants by Modular Arrays

Terry Beyer

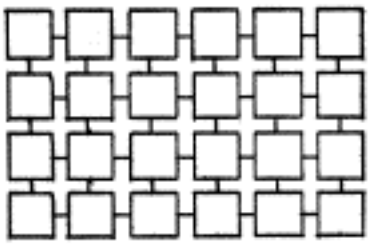
In this paper we study the recognition of topologically invariant properties of patterns by use of finite, rectangular 2-dimensional, iterative arrays of finite state automata (hereafter called modular arrays.) The use of modular arrays as pattern recognition devices has been studied by Atrubin [1] and by Unger [2].

Our aim is to show that modular arrays can not only recognize a large variety of topological invariants, but can do so in times that are almost minimal for a certain class of machines.

We begin by describing our model of the modular array as a pattern recognition device and presenting a simple but time consuming method of recognizing connectivity. Next we introduce a fundamental transformation of patterns and prove several interesting properties of the transformation. Finally, we apply the transformation to modular arrays to obtain fast methods of recognizing a wide variety of topological invariants.

Modular Arrays

Consider a finite, rectangular, 2-dimensional, iterative array of deterministic finite state automata. Such an array is pictured below with the automata represented by squares.



The automata used in such a construction are called modules and the entire arrangement is called a modular array. We assume that all of the modules in the array are identical and have been placed in the array with uniform orientation. Each module is connected directly to its four nearest neighbors. The array functions synchronously with the state of a module at time $t + 1$ being a function of the state of it and its four neighbors at time t.

The above description assumes that every module in the array has four nearest neighbors while in the diagram above it appears that the modules on the edges and corners of the array have fewer than four neighbors.

We assume that in fact these boundary modules do have four nearest neighbors where the neighbors not pictured above are special one state automata called edge markers. The use of edge markers allows a module to determine whether or not it is on the boundary of the array and to behave accordingly. The state transition diagram of a module thus explicitly contains descriptions of the behavior of that module in each of the sixteen possible situations in which it can find itself with respect to the edges of an array.

To operate a modular array as a recognition computer for black and white patterns, one designates two module states as initial states

corresponding to black and white. At time t = o every module in the array is placed in one of the two initial states and the pattern of states thus created represents the pattern to be processed. Beginning with the initial state representing the pattern, the array proceeds from state to state until finally a designated module in the array (we will always use the northwest corner module) enters one or the other of two specially designated decision states thus indicating whether the pattern is accepted or rejected. The decision states are assumed to be terminal in the sense that once a module enters such a state it remains in that state forever.

Remark, We always require the array to come to a definite accept or reject decision. This requirement is justified by the fact that given any module M with a designated accept state (but not necessarily a reject state) there is a module M* with an accept and a reject state such that the M* arrays will accept enactly those patterns accepted by the M arrays and reject enactly those patterns which are never accepted by the M arrays. (This result is most easily shown via the equivalence mentioned below between modular arrays and linear bounded automata although it can be proved directly and must be done so if timing considerations are important.)

Assume we have a predicate ψ defined on all finite, rectangular black and white patterns. A module M is said to recognize ψ if the modular arrays constructed from M accept enactly those patterns for which ψ is true and reject all others. The class of all predicates which are recognizable by modular arrays form a boolean algebra, contains all predicate which are true for only finitely many patterns, and is equal to the class of all predicates which are recognizable by 2-dimensional linear bounded automata.

Remark. A 2-dimensional linear-bounded automaton is a finite state deterministic automaton which is allowed to walk about on a pattern, read and write on the pattern with symbols from some finite alphabet

and sense the edges of the array. If the automaton eventually halts in an accepting state, it is said to accept the pattern, otherwise it is said to reject the pattern. It is not difficult to see that modular arrays can simulate linear bounded automata and vica versa. Using this equivalence one can show by a diagonalization argument that there are effectively recognizable predicates which cannot be recognized by any module.

Given a module M which recognizes a predicate ψ we will be interested in how "fast" M is able to carry out this recognition. Given any pattern P, let $t_M(P)$ be the amount of time required by an M-array to accept or reject P. The time $t_M(P)$ will vary from pattern to pattern, but in general one would expect $t_M(P)$ to increase as the number of squares in the pattern increases.
Let $f_M(n,m) = \max \{t_M(P) \mid P \text{ is an } n \times m \text{ pattern } \}$.
Then, f_M is called the guaranteed maximal time for M to recognize ψ. In the following section we will use the guaranteed maximal time as our criterion of how fast a given module recognizes a given predicate. Note that the amount of time required for a signal to travel from the southeast to the northwest corner of an n x m array is approximately n + m. Thus for reasonable predicates a guaranteed maximal time of approximately n + m is optimal.

We conclude this section by mentioning a result which we call the speed-up theorem. This result states that given any module M which recoginizes a predicate ψ, and given any real number $\varepsilon > 0$ exists a module M* such that

$$f_{M^*}(n,m) < (1 + \varepsilon)(m + n) + \varepsilon \cdot f_M(n,m)$$

for all n;m.

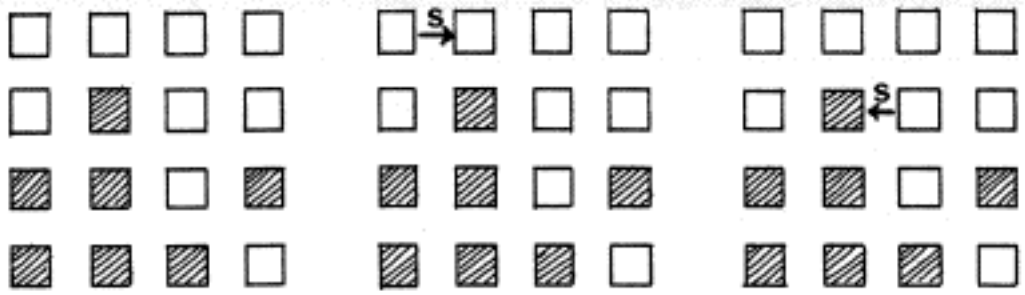
Sketch of Proof. The M* - array spends approximately $(1 - \varepsilon)(m + n)$ units of time packing the original pattern into the northwest portion of the array so that each module in that portion is responsible for a large block of squares in the original pattern. The next $2\varepsilon \cdot (n+m)$ units of time are used for an application of the firing squad problem. When the firing squad goes off, the packed portion of the M*-array begins simulating the action of the M-array on the pattern, but because of its packed nature it is able to do so at a higher rate of speed. A description of the firing squad problem can be found in Balzer [3] and the use of packing to speed up arrays is discussed in Cole [4].

A guaranteed maximal time of the form $f_M(n,m) = am + b\ n + c$ is said to be linear. A corollary of the speed up theorem is that if a predicate can be recognized with a linear guaranteed maximal time, then for any $\varepsilon>o$ it can be recognized with a guaranteed maximal time of $(1 + \varepsilon)(m + n)$. Therefore in view of our remarks above about $(m + n)$ being optimal (modulo a constant), we see that a linear guaranteed maximal time is "almost" optimal. The purpose of this paper is to show that a wide variety of topologically invariant predicates can be recognized with a linear guaranteed maximal time.

Recognition of Connectivity

We now describe a simple method of recognizing connectivity by modular arrays. That is, we wish to describe a module M which recognizes ψ-connected. Rather than describe M explicitly via a state diagram or state transition table, we will describe M implicitly by describing the action of an M-array on a typical pattern. It will be left to the reader to convince himself that one could write down a state description of a module M such that any M-array would carry out the process described.

Initially (t=o) the pattern is introduced into the array. The northwest corner module immediately emits a scanning signal s which begins to scan the array row by row in a back and forth manner until it encounters a black square. This stage of the process is illustrated in the following figures.

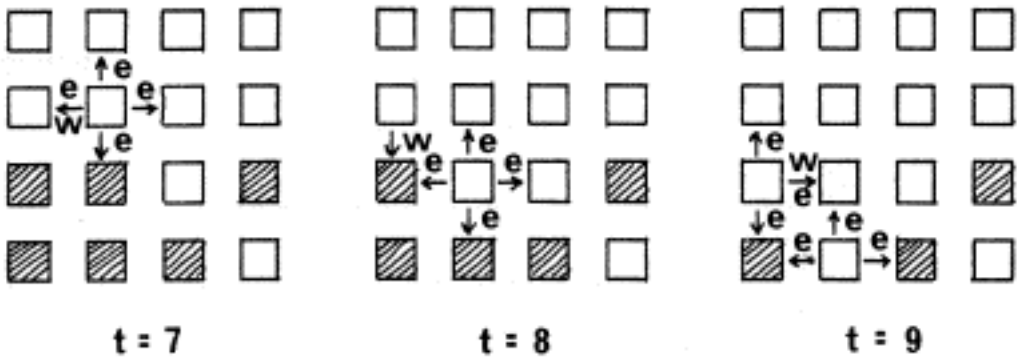


At the point at which s encounters the first black square two things happen. First of all a chain reaction of erasure is set off within the component of the pattern to which the black square belongs. The black square which was struck by s turns white and emits an erase signal e to each of its four neighbors. The e signals are ignored by

white squares but an e signal striking a black square causes it to turn white and emit e signals to its four neighbors.

In this manner the entire component is erased. The second thing which happens when s encounters the first black square is that s changes into a waiting signal w. The waiting signal continues the same zig-zag scanning motion which s had been using but does not interact with either black or white squares or with e signals which are propogating around the array in various directions. The w signal eventually completes the scan of the array and strikes one of the bottom corners of the array. At this point the erasure of the component is guaranteed to be complete. (Why?).

The first three stages of the erasure process are shown in the following figure.



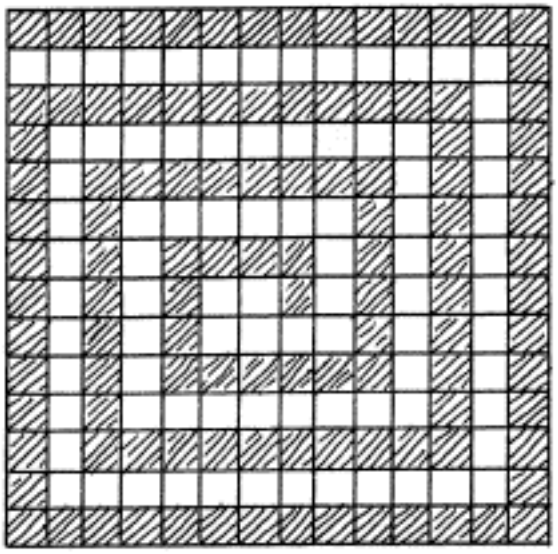
When w strikes the corner at the end of its scan, the pattern contains one less component than it did to begin with. All that remains to be done is to see if the remaining pattern is blank. This is accomplished by having the w signal rebound from the last corner as an accept signal a which scans up the array searching for a black square.

If a encounters a black square it is converted into a reject signal r which heads directly for the northwest corner to cause a reject. If a doesn't encounter a black square, it eventually strikes

the northwest corner causing an accept.

The case of the blank figure is handled by having the s signal rebound as a when it completes its scan.

The module implicitly described above recognizes connectivity with a guaranteed maximal time approximately 2nm and hence is not optimal. The reader is challenged to find a faster method of recognizing connectivity before reading the next section. By a faster method we mean of course a method with a linear guaranteed maximal time. A good (or bad, depending on your point of view) example to keep in mind while searching for a linear method is the pattern illustrated below which has length and area of about $\frac{1}{2}$ nm.



A Fundamental Transformation

In this section we present a simple transformation of black and white patterns which will have important applications to the recognition of topological invariants by modular arrays. The transformation will be studied in its own right in this section and its applications will be discussed in the following section.

We first introduce some notation and terminology. Let P be a pattern with n rows and m columns. We say P is an n x m pattern and introduce coordinates by numbering the rows from top to bottom, the columns from left to right and assigning the coordinate (i,j) to the square in the i=th row and j-th column. Two squares (i,j) and (p,q) are said to be adjacent if $|i-p| + |j-q| \leq 1$ and are said to be neighboring if $|i-p| < 1$ and $|j-q| < 1$. Two black squares are connected if there is a chain of pairwise adjacent black squares beginning with one and ending with the other. Two white squares are connected if there is a chain of pairwise neighboring white squares beginning with one and ending with the other. Note the asymmetric definition of connectedness for black and for white squares. Some such asymmetric definition is necessary if one is to retain such "nice" properties as the Jordan Curve Theorem.

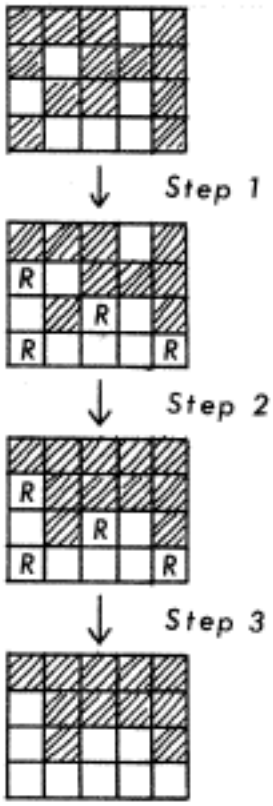
Remark. A notion of connectedness which is symmetric with respect to black and white can be obtained by assuming that each square "touches" all of the neighboring squares except the ones to the northeast and the southwest. This notion which is derived from a hexagonally partitioned pattern is, however, asymmetric with regard to direction.

The equivalence classes of black squares under the relation "connected" are called the components of P. The following definitions are motivated by our assumption that the pattern lies on a white background. The equivalence classes of white squares under the relation "connected" which do not contain squares on the border (that is squares in rows 1 or n or in columns 1 or n) are called holes. The remaining equivalence classes of white squares are lumped together into a class of

white squares called the background. A component or hole which contains only one square is said to be isolated, otherwise non-isolated.

Given a square in a pattern, the eight neighboring squares are referred to as the northern neighbor, the north-eastern neighbor, the eastern neighbor, and so on.

We now describe a transformation T which takes any n x m pattern P into a new n x m pattern T (P). The transformation may be thought of as taking place in three steps.



Step 1. Color all southeast corner squares of the black subpattern red. (That is if a square is black and its eastern and southern neighbors are white, color it red.)

Step 2. Color all southeast corner squares of the white subfigure black. (That is if a square is white and its eastern and southern neighbors are black and its southeastern neighbor is either red or black, color it black.)

Step 3. Color all red squares white.

We now informally describe the properties of T. The remainder of this section will be devoted to proving these properties. If one considers repeated applications of T to a pattern, one observes that each component is reduced to an isolated component which then disappears. Distinct components remain distinct and either vanish at different points or at the same point at different times. Similarly, each hole is reduced to an isolated hole which then vanishes with distinct holes remaining distinct and vanishing at different points or different times. It is easy to calculate exactly how many applications of T will be required to reduce a component or hole to a single square and exactly where that

square will be. The entire pattern, no matter how complex, will be reduced to the all white background in less than n + m applications of T.

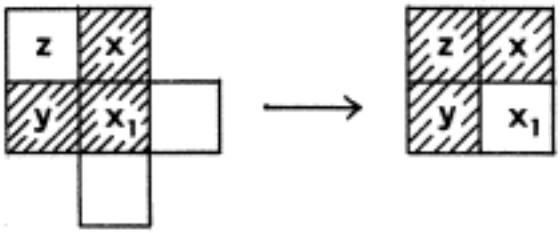
To begin proving the above statements we need some way of relating the components of P to those of T(P). This is done in the next three propositions by using the concept of a stationary point. A square X is called a stationary point of P if it is black in both P and T(P). Note that the stationary points are exactly those black squares in P which are not southeast corners of the black subpattern of P.

Proposition 1. Every non-isolated component of P contains a stationary point.

Proof: Let C be a non-isolated component and let X be a northwest corner of C. Then X must be a stationary point for otherwise it would also be a southeast corner and hence C = {X} would be isolated.

Proposition 2. Two stationary points are connected in P if and only if they are connected in T(P).

Proof: [==>] Let x and y be two stationary points of P which are connected. Then by definition there exists a sequence $x_0, x_1, \ldots, x_n$ of distinct pairwise adjacent black squares such that $x = x_0$ and $y = x_n$. We use induction on n. If n = 1 then x is adjacent to y and we are done. If n = 2 then either x_1 is also a stationary point in which case we are done, or x_1 is a southeast corner. In the latter case we have the situation depicted (next page), possibly with x and y interchanged, and one sees that the square z will be black T(P) no matter what its color in P. Thus x and y are connected in T(P).



P **T(P)**

Now assume $n \geq 3$. Observe that any chain of distinct pairwise adjacent black squares cannot contain two consecutive southeast corners. Thus either x_{n-1} or x_{n-2} is a stationary point and we may apply the induction hypothesis to the chain $x_o, \ldots, x_k$ and $x_k, \ldots, x_n$ where k is either $n - 1$ or $n - 2$. Thus x is connected to y in T(P) via x_k.

[$\Leftarrow$] Suppose x and y are connected in T(P) and let $x_o, x_1, \ldots, x_n$ be a sequence of pairwise adjacent black squares in P(T) such that $x = x_o$ and $y = x_n$. Again we use induction and again the cases for $n = 1$ and $n = 2$ with x_1 a stationary point (of P) are trivial, so assume $n = 2$ and x_1 is not a stationary point. Then x_1 must have been white in P since it is black in T(P). Hence x_1 must have satisfied the conditions in step 2 of the description of T and the situation depicted below must have obtained in P.

We know that x and y are adjacent to x_1 and are stationary points, so it will be sufficient to show that all stationary points which are adjacent to x_1 are connected to each other.

Now if a is a stationary point, then a is not a southeast corner in P and hence both a and a^1 are black in P. Thus a is connected to a^1. A similar argument holds for b. Thus all stationary points in P are connected to a^1. This completes the case for n = 2.

The remaining cases for $n \geq 3$ follow again from the observation that either x_{n-1} or x_{n-2} is a stationary point (although different reasoning must be used to make this observation now since $x_0, \ldots, x_n$ is a chain in T(P)).

Combining propositions 1 and 2 with the observation that every black square in T(P) is either a stationary point of P or is adjacent to a stationary point we have shown.

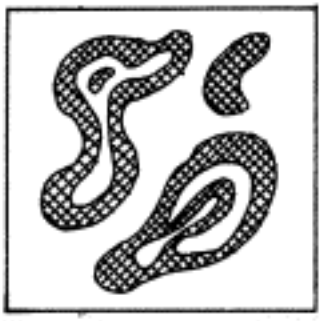
Proposition 3. There is a canonical 1 - 1 correspondence between the non-isolated components of P and the components of T(P).

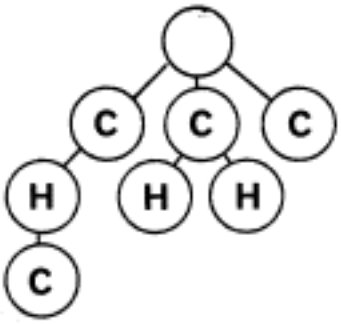
We now state without proof the corresponding proposition for holes which can be proved by methods similar to those above. However, a slightly different concept than that of stationary point must be used since some holes such as that illustrated below, have no stationary points.

P T(P)

Proposition 4. There is a cononical 1 - 1 correspondence between the non-isolate holes of P and the holes of T(P).

Given a pattern one can construct an associated tree which represents the containment relationships between the background, the components and the holes. A pattern and its associated tree are shown below.





Two patterns which have isomorphic trees are said to be topologically equivalent. The following proposition should be obvious by now.

Proposition 5. If P contains no isolated components or holes, then P and T(P) are topologically equivalent.

We now show how to compute the number of applications of T required to reduce a component to a single square and where that square will lie. Identical results can be proved for holes using similar arguments.

Given a component C of a pattern P, let

$T^0(C) = C$

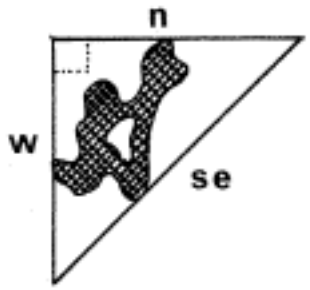
$T^k(C) =$ the canonical image of $T^{k-1}(C)$ under T for $k > 0$ (provided it exists)

$n(C) = \min \{i \mid$ row i intersects $C\}$

$w(C) = \min \{j \mid$ column j intersects $C\}$

$Se(C) = \max \{i+j \mid (i,j) \in C\}$

Note that n(C), w(C), and se(C) represent three lines forming a triangle such that C lies within the triangle and touches each line as shown on next page.



We will show that the component vanishes at the square indicated by the dotted lines and that the number of applications of T required to achieve this is proportional to the distance from this square to the se line.

Proposition 6. If C is a non-isolated component, then

$$n(T(C)) = n(C)$$
$$w(T(C)) = w(C)$$
$$se(T(C)) = se(C)-1$$

Proof: [$n(T(C)) = n(C)$] It is clear that $n(T(C)) \geq n(C)$ since each black square in T(C) is either a stationary point (of P) or is the western neighbor of a stationary point. On the other hand, if (i,j) is the western most point of C which lies in row n(C), then (i,j) must be a stationary point and hence $n(T(C)) \leq n(C)$.
[$w(T(C)) = w(C)$]. This result follows immediately from the above and the northwest symmetry of T.
[$se(T(C)) = se(C)$]. Any square (i,j) in C such that $i + j = se(C)$ must be a south east corner of C and hence is adjacent to a stationary point (p,q) such that $p+q = se(C) - 1$. Thus $se(T(C)) \geq se(C)-1$. On the other all such squares (i,j) do not appear in T(C), so $se(TC)) \leq se(C)-1$.

Proposition 7. If C is a non-isolated component and $k(C) = se(C) - n(C) - w(C)$, then $T^{k(C)}(C)$ is an isolated component located at $(n(C), w(C))$.

Proof: By proposition 6 we have $k(T(C)) = k(C)-1$.
Thus by induction $k(T^{k(C)}(C)) = k(C) - k(C) = 0$
which can only hold for an isolated component.
That component must be located at $(n(T^{k(C)}(C)),$
$w(T^{k(C)}(C))) = (n(C), w(C))$ again by proposition 6.

Corollary: If P is an nxm pattern, then $T^{n+m-1}(P)$ is the all white figure.

Proof: Apply Proposition 7 and the fact that $k(C)<n+m-2$ for any component C of P.

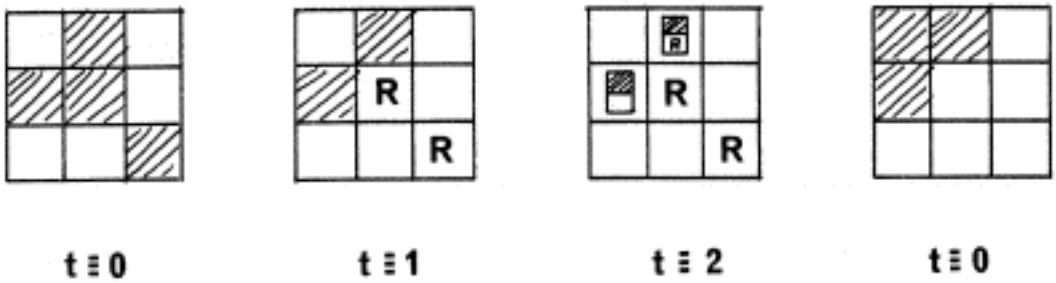
Fast Recognition of Topological Invariants.

The transformation, T, described in the previous section forms the basis of the recognition schemes to be presented in this section. These recognition schemes all have linear guaranteed maximal times and hence by the remarks above can be considered near optimal methods.

In all of the recognition schemes described below, the modular array is thought of as consisting of two layers, a lower, transormation layer which carries out successive transformations of the initial pattern, and an upper, observation layer which watches the transformations taking place, gathers and processes information, and finally comes to a decision about the pattern.

The transformation can be carried out in the lower layer of the array at the rate of one transformation every three units of time, thus becoming dormant after 3 (m + n - 1) units of time (if not sooner).

At time $t \equiv o \pmod 3$ the pattern is represented in the lower layer. By time $t \equiv 1$ those squares which are to turn red have done so. By time $t \equiv 2$ each black square has entered a state which not only indicates that it is black but also indicates what state its southern neighbor is in, thus making available to each white square the necessary information for the final step. By the $t \equiv 3 \equiv o$ the transformation is complete. The process is illustrated below.



It now remains to describe the observation layer for each predicate to be recognized. The observation layers all watch for the disappearance of components or holes in the lower layer and generate appropriate signals at each such disappearance. These signals are then processed and a decision is reached. In some cases it is necessary for the northwest corner module to know that it has received all the information required for a decision. In these cases the southeast corner module sends out a timing signal which propogates through the array at an appropriate rate. When the timing signals reaches the northwest corner module, all other signals must have preceded it.

We now list some specific predicate which can be recognized in this manner. "The pattern is connected": Signals are only generated by vanishing components. As each signal is generated it heads for the northwest module. The figure is rejected if more than one such signal is received.

"All components are simply connected": Apply the method above to holes rather than components and reject any figure with one or more holes.

The two predicates above are examples of the general predicate. "The pattern contains at least i and no more than j components and at least k and no more than l holes." This predicate is easily recognized for any $0 \leq i,j,k,l < \infty$ by simple modification of the above techniques.

Now consider the predicate, "no component contains more that one hole." This predicate is not in the above form but may be recognized by having the observation layer keep each hole signal positioned above the component in which the hole was located. If a component contains two or more holes, the hole signals must eventaully bump into each other as the component is reduced to a single square. Two bumping holes cause a reject signal. A hole signal located over a component vanishes when the component vanishes.

The technique of keeping signal positioned above components or holes gives rise to a plethora of predicates, one of which is "every component which is contained in a hole in another component is simply connected." By building a large enough signal set in the observation layer one can recognize any predicate of the form. "The pattern is toplogically equivalent to the pattern P" for any fixed pattern P.

References

1. Atrubin, A.J.
"A Study of Several Planar Iterative Switching Circuits." S.M. Thesis, MIT, E.E. Department, 1968.

2. Unger, Stephen H.
"Pattern Recognition Using Two-Dimensional Bilateral, Iterative Combinatorial Switching Circuits." Proceedings of the Symposium on Mathematical Theory of Automata, ed. by J. Fox, Polytechnic Press, 1963, pp. 577 - 592.

3. Balzer, Robert.
"An 8-state minimal time solution to the firing squad synchronization problem." Information and Control 10 1967, pp. 22 - 42.

4. Cole, Stephen N.
"Real-time Computation by Interactive Arrays of Finite-State Machines." Harvard University, Division of Engineering and Applied Physics. Ph.D. Thesis, 1964.