MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence Memo. 135 June 1967

Automata On A 2-Dimensional Tape

M. Blum and C. Hewitt

This paper explains our approach to the problem of pattern recognition by serial computer. The rudimentary theory of vision presented here lies within the framework of automata theory. Our goal is to classify the types of patterns that can be recognized by an automaton that scans a finite 2-dimensional tape. For example, we would like to know if an automaton can decide whether or not a given pattern on a tape forms a connected region.
This paper should be viewed as a Progress Report on work done to date. Our goal now is to generalize the theory presented here and make it applicable to a wide variety of pattern-recognizing machines.

1. The finite automata we consider are free to scan the tape horizontally and vertically. The tape itself is a finite square ruled horizontally and vertically into ε-squares, i.e., squares of sidelength ε. The bounding ε-squares of the tape are marked with the special symbol B (border), while the interior squares are marked with either a 0 (white) or 1 (black):

	c_1	c_2	c_3	c_4		
	B	B	B	B	B	B
r_1	B	0	0	1	0	B
r_2	B	0	1	1	1	B
r_3	B	0	0	0	1	B
r_4	B	1	1	1	0	B
	B	B	B	B	B	B

Example of a Tape

All rows and columns except for the border rows and columns are labeled $r_1, \ldots, r_m$ and $c_1, \ldots, c_m$, respectively. The automaton begins its scan on square (r_1, c_1) and continues the scan until it either accepts or rejects the tape, or else cycles. The automaton is not allowed to fall off the tape. At each moment of time, the automaton scans exactly one square, and shifts its scan by one square north, east, south, or west, depending on its internal state and on the symbol that appears on the scanned square. The automaton can not write on its tape; it can only scan it.

Formally, a 2-dimensional automaton is a system $\mathfrak{A} = (S, f, g, s_0, s_1, s_2)$, where S is a finite set, the set of states. f is the "next state" function mapping $S-\{s_1, s_2\} \times \{0, 1, B\}$ into S. g is the "direction of motion" function mapping $S-\{s_1, s_2\} \times \{0, 1, B\}$ into {N, E, S, W}.

s_0 is the initial state, s_1 is the accepting state, and s_2 is the rejecting state. We say that $\mathfrak{A}$ accepts a tape if, by starting $\mathfrak{A}$ in state s_0 on the square (r_1, c_1), $\mathfrak{A}$ eventually halts in state s_1; $\mathfrak{A}$ rejects the tape if it halts in state s_2. We say that $\mathfrak{A}$ can decide whether or not the pattern on a tape has property P if it accepts all tapes with the property P and rejects all tapes that are without it.

Examples. The following properties of tapes are decidable by automata:

1. The tape contains precisely k 1's. To decide whether a tape is of this kind, the automaton scans the tape and counts 1's.

2. The tape is white except for a single rectangle (i.e., the 1's on the tape form a filled-in rectangle) with sides parallel to the sides of the tape. The automaton decides whether a tape has such a pattern on it by scanning the columns of the tape one after the other, starting with c_1 and ending with c_m. Each time the automaton passes through a boundary of the inscribed rectangle, it checks the slope of the boundary at that point by scanning a neighborhood of the point: It makes sure that the slope is zero except when the scan is at the left or right edge of the rectangle.

3. The tape contains a single square with sides parallel to the sides of the tape. Once the automaton has checked that the pattern is a rectangle, it can check whether or not it is a square by finding a vertex of the rectangle and then scanning from that vertex at 45° to the vertical toward the other vertex. If the opposite vertex is reached, then the rectangle is a square, otherwise it is not.

4. At least one of any number of pathwise-connected components on the tape is square: It is easy to see how the automaton can systematically check each of the individual components without cycling. Each of the components must be checked in such a way that if the component is not a square, then the automaton can return to the point of entry into that component.

A property that is not decidable by an automaton is whether a tape is symmetrical about the central column. This fact can be proved by the methods used to prove theorem 1.

Many problems about finite automata that are easy to solve in the 1-dimensional case are impossible in the 2-dimensional case. For example, the problem whether a 1-dimensional automaton accepts a tape is solvable by an effective algorithm, while the problem whether a 2-dimensional automaton accepts even a blank tape is recursively unsolvable. This follows immediately from the fact that a 2-counter automaton is universal (cf. Minsky): Hence, the x, y coordinates of the automaton on the 2-dimensional tape serve in place of the counters.

In the remainder of this paper, we use the word "automaton" to mean 2-dimensional automaton.

2. The theorem in this section provides an example of a problem that an automaton cannot solve. The proof of the theorem is particularly important because it embodies an idea that reappears in the proofs of several other theorems. This is the idea of two chunks of tape being indistinguishable to the automaton.

Definition. A chunk of tape is a borderless subsquare of the tape with some fixed pattern of 0's and 1's on it. The length x of the side of the chunk is called the sidelength of the chunk. We suppose that the squares forming the boundary of a chunk are numbered x_1, x_2, ..., around the perimeter of the tape.

Definition. We say that 2 chunks of equal sidelength are $\mathfrak{A}$-equivalent if the following holds: Let x_i be any point on the boundary of a chunk, and let s_i be any state of $\mathfrak{A}$. Suppose that if $\mathfrak{A}$ enters one of the chunks at x_i and in state s_i, it exits that chunk at some point x_j and in some state s_j. Then if $\mathfrak{A}$ enters the other chunk at x_i and in state s_i, it also exits that chunk at x_j and in state s_j.

Thus if 2 chunks are $\mathfrak{A}$-equivalent, we say $\mathfrak{A}$ cannot distinguish between the patterns on those chunks. Clearly, $\mathfrak{A}$-equivalence is an equivalence relation on chunks.

Theorem 1. In general an automaton cannot decide whether a tape whose side has odd length has a 1 in its center square.

Proof We assume that an automaton, $\mathcal{A}$, can make this decision. We also suppose, without loss of generality, that when $\mathcal{A}$ halts, it does so on square (r_1, c_1). A chunk of tape of sidelength x has 4x squares along its perimeter, so for each entry into that chunk, an automaton with n states can exit in one of 4xn ways, or else not exit at all: a total of 4xn + 1 possibilities. Since an automaton has 4xn ways to enter a chunk, it follows that there are at most $(4xn + 1)^{4xn}$ $\mathcal{A}$-equivalence classes of chunks of sidelength x. Thus $\mathcal{A}$ can distinguish among at most this many chunks. However, the total number of chunks of sidelength x is 2^{x^2}. Since $2^{x^2} > (4xn + 1)^{4xn}$ for $x \gg n$, it follows that there are at least 2 different $\mathcal{A}$-equivalent chunks. These 2 chunks differ, say, in square "sq". Construct two tapes to contain these chunks, and make the tapes sufficiently large so that the square sq appears in the center of both tapes: Now notice that $\mathcal{A}$ cannot distinguish between these two tapes. Hence $\mathcal{A}$ cannot decide whether the center of a square tape contains a 1 or a 0. QED

The idea of a nondeterministic 1-dimensional automaton (cf. Rabin & Scott) extends naturally to that of a nondeterministic 2-dimensional automaton.

Corollary 1 A nondeterministic automaton is more powerful than a deterministic automaton.

Proof A nondeterministic automaton can decide whether a square tape contains a 1 or a 0 in its center square. It does this by initiating a scan from (r_1, c_1) along the diagonal to (r_m, c_m). Whenever the automaton sees a 1 along this diagonal, it may make a 90° left turn, and move toward the border, or continue along its way to (r_m, c_m). If it can reach (r_1, c_m) by making the correct 90° turn, then the tape contains a 1 in its center square. Otherwise the tape does not.

QED

3. We extend the power of an automaton by giving it a finite number of markers, labeled $m_1, \ldots, m_k$. At any moment the automaton may place a marker m_i on the particular square of the tape it is scanning, and at that moment any other occurrence of m_i on its tape instantly disappears. We call this marker an "abstract marker". Another kind of marker, also denoted by $m_1, \ldots, m_k$, is called the "physical marker": The physical marker is a kind of labeled pebble that an automaton moves about on the tape. To get the physical marker transferred from one position to another, the automaton must actually go to the marker and move it to its new position.

Both abstract and physical markers may be stacked like poker chips on a single square. Actually, it is easy to prove that they need not be:

Theorem 2.1 k-marker automata that can place at most one marker on any ϵ-square are just as powerful as k-marker automata that can stack any number of these markers on an ϵ-square.

Obviously, an automaton with k abstract markers can simulate one with k physical markers. The reverse is also true, as we shall see. In this paper, depending on the theorem we wish to prove, it is sometimes convenient to switch from one type of marker to another.

Theorem 2.2 An automaton with k abstract markers can be simulated by one with k physical markers.

Proof Let $\mathcal{A}$ denote the automaton with abstract markers, and let $\mathcal{P}$ denote the automaton with physical markers. $\mathcal{P}$ simulates $\mathcal{A}$ by moving about on the tape, placing markers just as $\mathcal{A}$ would. (This is okay until $\mathcal{A}$ is required to place a marker than appears elsewhere.)

During this simulation, $\mathcal{P}$ remembers the marker, call it m_i, that it last saw and the state, call it s_i, it was in when it last saw m_i. Now when $\mathcal{A}$ is required to put down a marker, m_d, that has been placed

elsewhere, $\mathcal{P}$ achieves this same result by the following roundabout method: $\mathcal{P}$ first locates m_d by scanning the tape. $\mathcal{P}$ picks up m_d and carries it as it scans for m_i. When $\mathcal{P}$ finds m_i, it recalls what state it was in when it last left m_i, enters that state, and proceeds with the simulation. QED

We have mentioned that the markers are labeled 1 through k. Actually, this is unnecessary:

Theorem 2.3 An automaton with unlabeled markers can simulate one with labeled markers.

The automaton does so by keeping track of the positions of the markers relative to one another.

An automaton with 1 marker is more powerful than an automaton without any markers. For one thing, the automaton with a marker can decide whether the center square of a tape contains a 1 or a 0: The automaton starting at (r_1, c_1) moves its marker along the diagonal toward (r_n, c_n). Each time it moves its marker one more square along the diagonal, the automaton drops the marker and runs off at 90° to the diagonal looking for the tape square (r_1, c_n). When it finds that square, the marker is on the center square, and so the automaton can make its decision.

Theorem 3 Automata with $2k + 4$ markers, $k \geq 0$, are more powerful than automata with k markers: There is a certain property P such that a $2k + 4$ marker automaton can decide whether or not a tape has the property, but no k-marker automaton can make this decision.

Observe that it is possible to represent the state diagram of any k-marker automaton on a tape. This can be so formalized that a 0-marker automaton can decide whether or not a tape contains the representation of some k-marker automaton. We suppose that such a representation of automata has been formalized, and we let $t(\mathcal{B})$ be the tape description of the state diagram of an automaton $\mathcal{B}$. We note the somewhat curious fact that any $\mathcal{D}$ may be required to decide about $t(\mathcal{B})$, and that $\mathcal{B}$ must accept, reject or cycle on that tape.

Outline of a Proof Let $\mathcal{B}$ be any k-marker automaton. Let P be the property:
(1) The tape is of type t ($\mathcal{B}$).
(2) The sidelength x of the tape is greater than the number, n, of states of $\mathcal{B}$.

(3) $\mathcal{B}$ does not accept this tape.
$\mathcal{U}$ is a 2k + 4 marker automaton that accepts tapes with property P and rejects all others. $\mathcal{U}$ is described as follows: It uses marker m_1 to keep track of the state of $\mathcal{B}$ as $\mathcal{B}$ scans t($\mathcal{B}$). The marker m_2 is used to keep track of the position of $\mathcal{B}$ as $\mathcal{B}$ scans t($\mathcal{B}$). The k markers $m_3, \ldots, m_{k+2}$ are used to represent $\mathcal{B}$'s k markers. The remaining k + 2 markers are used to tell whether or not $\mathcal{B}$ is cycling; these markers are used for counting the number of steps taken by $\mathcal{B}$. They are started together on (r_1, c_1), and they are moved in such a way that, if $\mathcal{B}$ cycles, they eventually appear in all possible combinations of positions on the tape. In this case, the markers can be made to end up together on (r_m, c_m), and this is the only case in which they do. Thus if all the markers reach (r_m, c_m), $\mathcal{U}$ concludes that $\mathcal{B}$ cycles on t($\mathcal{B}$) and so $\mathcal{U}$ accepts t($\mathcal{B}$).

On a tape of sidelength x, k + 2 markers can assume $(x^2)^{k+2}$ different positions. Thus $\mathcal{U}$ is able to simulate up to $(x^2)^{k+2}$ moves of $\mathcal{B}$. But $\mathcal{B}$ has n states, so it can make at most $(x^2)^k \cdot x^2 \cdot n$ moves without cycling. For large x, $(x^2)^{k+2} > (x^2)^k \cdot x^2 \cdot n$. Therefore there exists a tape t($\mathcal{B}$) of sufficiently large sidelength x such that $\mathcal{U}$ accepts t($\mathcal{B}$) if and only if $\mathcal{B}$ does not. QED

4. Definition The pattern on a tape is the set of all black squares that appear there. Two squares are adjacent if they share a common edge (not just a common vertex). The boundary of a pattern is the set of all black squares that are adjacent to white squares. A pattern is connected if and only if any 2 black squares are joined by a string of adjacent black squares.

Although it seems certain that a 0-marker automaton cannot decide if a pattern is pathwise-connected, we have no proof of this result. On the positive side, we can prove

Theorem 4 A 1-marker automaton, $\mathfrak{A}$, can decide if a pattern is pathwise-connected.

Lemma 1 Let R be a pathwise-connected pattern, and let p_1, p_2 be two boundary points of R. Suppose a curve joins p_1 and p_2 without intersecting R at any other points. Then there exists a curve joining p_1 and p_2 that lies entirely on the boundary of R.

Definition We say that a column cuts a pattern in two if and only if a black square lies to the right of the column, and a black square lies to the left and no black square that lies to the right is connected by a string of adjacent black squares to a black square that lies to the left of the column. Note that the cutting column may contain black squares.

Lemma 2 A pattern R is pathwise-connected if and only if (1) No column cuts the pattern in two, and (2) Any pair of boundary squares that lie in the same column and have only white squares lying between them are connected.

Proof of Theorem $\mathfrak{A}$ checks conditions (1) and (2) of lemma 2:

(1) $\mathfrak{A}$ scans the tape column by column from c_1 to c_m to decide whether a column cuts the pattern in two. If $\mathfrak{A}$ finds such a column, it announces that R is not connected. Otherwise, $\mathfrak{A}$ proceeds with (2).

(2) $\mathfrak{A}$ scans the columns from c_1 to c_m. Each c_i is scanned from (r_1, c_i) to (r_m, c_i), and this scanning is interrupted whenever $\mathfrak{A}$, during its southward movement within c_i, leaves R at some point (r_k, c_i) and re-enters it at some point (r_e, c_i), $e > k$. When this happens, $\mathfrak{A}$ leaves the

marker at the point (r_e, c_i) where it re-entered R. Then, starting at that point, $\mathcal{A}$ travels around the boundary containing that marker. Eventually, one of two things happens: (1) $\mathcal{A}$ finds itself at the point (r_k, c_i) on the R-boundary directly above the marker. In this case, the point (r_e, c_i) on which the marker lies is connected to the point (r_k, c_i) above it. In this case, $\mathcal{A}$ continues the vertical scanning interrupted above. (2) $\mathcal{A}$ returns to the marker without passing through the point (r_k, c_i). In this case, we know from lemma 1 that the pattern is not pathwise-connected.

If $\mathcal{A}$ scans the whole tape without finding that R is disconnected, it announces that R is connected. QED

Corollary A 1-marker automaton can decide whether a given pattern is simply connected.

Proof The idea is to have the automaton check that the pattern R and its complement are pathwise-connected. This is complicated by the fact that R may split the tape into several disjoint components. The details of what the automaton must do in this case are left to the reader. QED

5. In this section we study the problem of deciding whether one region is a translation of another. As a first example, we note that a 1-marker automaton can decide whether a square region is a translation of another. As we shall see, however, a 1-marker automaton cannot decide whether a simply-connected region is a translation of another.

We extend our definition of $\mathcal{U}$-equivalence to the case where $\mathcal{A}$ has physical markers.

Definition We say that two chunks of equal sidelength are $\mathcal{A}$-equivalent if they are $\mathcal{U}'$-equivalent, where $\mathcal{U}'$ is gotten from the automaton $\mathcal{A}$ by taking away the markers from $\mathcal{A}$. (We realize that this definition is rather informal).

Theorem 5 Suppose that $\mathfrak{A}$, a 1-marker automaton, is presented with a tape that contains exactly two disjoint simply-connected regions. Then $\mathfrak{A}$ cannot decide whether one of these regions, R_1, is a translation of the other, R_2.

Definition The pattern on a chunk is the set of all black squares on that chunk.

Lemma The number of different simply-connected patterns that fit in a square of sidelength x is at least $2^{(\frac{x}{2}-2)\cdot\frac{x}{2}} \approx 2^{x^2/4}$.

Proof of Lemma The square on the right contains a simply-connected Pattern and $(x-2)\cdot\frac{x}{2}$ checkmarked unit-squares. Any subset of these checkmarked unit-squares may be filled in, and the resulting Pattern will still be simply-connected. The result follows. QED

Proof of Theorem Actually, we prove the somewhat stronger result that an automaton cannot decide whether R_1 is a translation of R_2, given that R_1 is restricted to the left half (LH) of the tape while R_2 is restricted to the right half (RH).

We assume to the contrary that the automaton $\mathfrak{A}$ can decide. We further assume without loss of generality that (1) $\mathfrak{A}$ has a physical marker, and (2) $\mathfrak{A}$ ends up with the marker at the top of the central column in case it decides 'yes', but with the marker at the bottom of the central column in case it decides 'no'.

The idea of the proof is to show that there exist different simply-connected regions R_1 & R_2 s.t. $\mathfrak{A}$ must carry its marker back and forth across the central column in the same way when both LH and RH contain R_1 as when LH contains R_1 and RH contains R_2.

Suppose the tape has sidelength ℓ. Consider chunks of sidelength $x = \ell/2$. These chunks may contain any one of $2^{x^2/4}$ simply-connected patterns. If $\mathfrak{A}$ has n states, there are $(4xn)^{4xn}$ $\mathfrak{A}$-equivalent classes of chunks.

The largest such class has at least $y = \frac{2^{x^2/4}}{(4xn)^{4xn}} = \frac{2^{\ell^2/16}}{(2\ell n)^{2\ell n}}$ different $\mathcal{U}$-equivalent simply-connected regions.

To begin, we suppose the tape has any one of these y regions in LH and any one in RH. We partition the y different $\mathcal{U}$-equivalent simply-connected regions which appear in LH into ℓn $\mathcal{U}_1$-equivalence classes: First note that if $\mathcal{U}$ is started with its marker in LH, then $\mathcal{U}$ carries its marker into RH in one of ℓn ways ($\mathcal{U}$ must carry the marker across the central column in order to decide which one of the y regions appears in RH). Now say any two $\mathcal{U}$-equivalent regions are $\mathcal{U}_1$-equivalent provided that, no matter which one of these particular 2 regions appears in LH (and no matter which of the y $\mathcal{U}$-equivalent regions appears in RH), $\mathcal{U}$ carries its marker across the central column in the same way for both regions.

At least one of these $\mathcal{U}_1$-equivalence classes has $y_1 = y/\ell n$ members. Now suppose the tape has any one of these y_1 regions in LH and anyone in RH. The argument continues now with $\mathcal{U}$'s marker being in RH. $\mathcal{U}$ may move across the central column without its marker any number of times. However, $\mathcal{U}$ must eventually carry the marker across into LH if it is to distinguish which of the y_1 (different $\mathcal{U}$-equivalent) simply-connected regions appears in LH. $\mathcal{U}$ can carry the marker across in one of ℓn-1 ways. By the same argument as above, at least $y_2 = y/\ell n(\ell n-1)$ of these regions are $\mathcal{U}_2$-equivalent in the sense that no matter which one of these y_2 regions appears in RH (and no matter which of the y_1 regions appears in LH) the automaton moves its marker across the central column from RH to LH in the same way.

Continuing in this way, one finds that there are $y/(\ell n)!$ regions which are $\mathcal{U}_{(\ell n)!}$-equivalent in the sense that they are indistinguishable to an $\mathcal{U}$ that carries its marker across the central column at most $(\ell n)!$ times. But $\mathcal{U}$ cannot carry its marker across the central column more than $(\ell n)!$ times without cycling. But $y/(\ell n)! \gg 1$ for $\ell \gg n$. Hence $\mathcal{U}$ cannot decide if the region in RH is really a translation of the one in LH. QED

It is trivial to show that a 2-marker automaton can decide whether one

simply-connected region is a translation of another. Hence we have

Corollary 1 A 2-marker automaton is more powerful than a 1-marker automaton.

It is considerably more difficult to prove that a 2-marker automaton can decide whether an arbitrary pathwise-connected region is a translation of another. The following proof was suggested by Mr. Terry Beyer. It is a considerably simpler version of our original proof.

Theorem 6. Suppose a tape contains two disjoint pathwise-connected regions, R_1 and R_2. An automaton with two markers can decide whether or not R_1 is a translation of R_2.

Proof We shall give a set of instructions for the automaton, and prove that these instructions do the job. Here is a general outline:

(1)-(3) $\mathfrak{A}$ puts m_1 on an outer boundary point, p_1, of R_1, and $\mathfrak{A}$ puts m_2 on an outer boundary point, p_2, of R_2, which is chosen so that if $R_2 = T(R_1)$, then $p_2 = T(p_1)$.

(4) $\mathfrak{A}$ moves m_1 about the outer boundary of R_1 and m_2 about the outer boundary of R_2 and checks that these boundaries are alike.

(5)-(6) $\mathfrak{A}$ scans and compares the interiors of R_1 and R_2, checking that these interiors are alike.

This completes the proof.

We fill in this outline now, because it is non-trivial to show that $\mathfrak{A}$ can really do these things.

(1) A standard way for an automaton to find a point p_1 on the outer boundary of a component R_1 is to have it scan the tape column by column, from c_1 to c_m, and scan each c_i from (r_1, c_i) to (r_m, c_i) until it finds the first shaded square, p_1. Note that $\mathcal{A}$ can always find this p_1 no matter where $\mathcal{A}$ may be, by going to (r_1, c_1) and scanning as above. To find a point of R_2, $\mathcal{A}$ scans the tape in reverse from c_m to c_1, scanning c_i from (r_m, c_i) to (r_1, c_i). If $R_2 = T(R_1)$, then the point that $\mathcal{A}$ finds must be a point, q_2, of R_2. However, q_2 may be a point of R_1 if $R_2 \neq T(R_1)$. Since $\mathcal{A}$ does not know whether $R_2 = T(R_1)$, it puts m_1 on p_1, m_2 on q_2 and proceeds in (2) to check whether $q_2 \in R_1$.

(2) $\mathcal{A}$ checks whether $q_2 \in R_1$ by scanning around the boundary that contains m_2. If $\mathcal{A}$ finds m_1 then $q_2 \in R_1$. If $\mathcal{A}$ returns to m_2 without finding m_1 then $q_2 \in R_2$. If $q_2 \in R_1$, $\mathcal{A}$ answers that $R_1 \neq T(R_2)$. If $q_2 \in R_2$, $\mathcal{A}$ proceeds as in (3).

(3) $\mathcal{A}$ finds the point $p_2 \in R_2$ that corresponds to $p_1 \in R_1$ in the sense that if $R_2 = T(R_1)$, then $p_2 = T(p_1)$. To do this, $\mathcal{A}$ finds the easternmost point of R_2, and if there are several such points, it finds the northernmost one among them. This point is p_2. (We have omitted some detail of how $\mathcal{A}$ does this: It finds the point $p_2 \in R_2$ by placing both m_1 and m_2 on the outer boundary of R_2, and shuffling them about on that boundary. $\mathcal{A}$ keeps m_1 on the easternmost point it has so far found on R_2. The marker m_2 is moved about on the same boundary of R_2, and each time it is moved, $\mathcal{A}$ leaves it and scans the tape looking for m_1. If m_2 is east or north of m_1, $\mathcal{A}$ puts m_1 in place of m_2 and continues to move m_2 about on the boundary of R_2. $\mathcal{A}$ knows that m_1 is at point p_2 when $\mathcal{A}$ has moved m_2 around the boundary from m_1 back to m_1 without shifting the position of m_1.)

(4) In (3), $\mathcal{A}$ placed m_1 on p_1 and m_2 on p_2. Now $\mathcal{A}$ compares the outer boundaries of R_1 and R_2 by moving m_1 on the boundary of R_1 in unison with

m_2 on the boundary of R_2. ($\mathcal{H}$ can tell when the markers have returned to the starting points p_1 and p_2: Each time $\mathcal{H}$ moves the markers, $\mathcal{H}$ goes to (r_1, c_1), scans for p_1, and checks whether m_1 is on p_1. If m_1 is not on p_1, $\mathcal{H}$ continues to move m_1 and m_2 on their respective boundaries). (5) $\mathcal{H}$ next uses the markers to scan and compare the interior of R_1 with that of R_2. $\mathcal{H}$ does this by moving m_1 vertically through R_1 and m_2 in unison through R_2. Each move of m_1 is from an outer boundary of R_1 (across any number of inner boundaries) to the next outer boundary of R_1. By completely scanning R_1 and R_2, $\mathcal{H}$ decides whether or not they are alike. To be explicit, suppose m_1 is on an outer boundary point p of R_1 and m_2 is on the corresponding point T(p) of R_2. $\mathcal{H}$ moves m_1 south through R_1 until it reaches a boundary point. It drops m_1 at that point, and it drops m_2 on the corresponding point of R_2. Then $\mathcal{H}$ determines whether m_1 is on an outer boundary of R_1. ($\mathcal{H}$ does this by moving along the boundary that contains m_1, checking at each point whether a shaded square lies on the same row to the east. If a shaded square appears east of each point on that boundary, then m_1 lies on an inner boundary of R_1. If a border square appears east of some point, then m_1 lies on the outer boundary of R_1). If m_1 is not on the outer boundary, $\mathcal{H}$ continues the scan with m_1 and m_2, checking whether both regions are alike, until m_1 does get placed on the outer boundary. If m_1 does lie on the outer boundary of R_1, m_2 must be on the outer boundary of R_2, because the outer boundaries of R_1 and R_2 are identical. Finally, $\mathcal{H}$ shifts m_1 and m_2 north and places them on the first outer boundary point it finds, the ones whence it came. (6) $\mathcal{H}$ shifts from one point on the boundary to the next adjacent one. It is easy to see how $\mathcal{H}$ does this. Then $\mathcal{H}$ continues at (5).

By the procedure described above, m_1 scans all of R_1 and m_2 simultaneously scans all of R_2, so R_1 and R_2 are properly compared. QED

BIBLIOGRAPHY

Minsky, Marvin, "Computation: Finite and Infinite Machines," Prentice Hall, 1967.

Minsky, Marvin, and Papert, Seymour, "Linearly Unrecognizable Patterns," Proc. of Symposia in Applied Math., Vol. 19, ed. Schwartz, to appear.

Rabin, Michael O. and Scott, Dana, "Finite Automata and Their Decision Problems," IBM Journ. of Res. and Dev., 3, No.2, (April '59)114-125.