Rectangles and squares recognized by two-dimensional automata

Jarkko Kari[†1] and Cristopher Moore[‡]

[†] *Department of Computer Science, 15 MLH, University of Iowa, Iowa City IA 52242 USA,* jjkari@cs.uiowa.edu

[‡] *Computer Science Department and Department of Physics and Astronomy, University of New Mexico, Albuquerque NM 87131, and the Santa Fe Institute, 1399 Hyde Park Road, Santa Fe NM 87501,* moore@cs.unm.edu

We consider sets of rectangles and squares recognized by deterministic and non-deterministic two-dimensional finite-state automata. We show that NFAs are strictly more powerful than DFAs, even for pictures over a one-symbol alphabet. In the process, we show that the picture languages recognized by NFAs are not closed under complement, resolving a long-standing open question. We also show that NFAs can only recognize sets of rectangles from the outside that correspond to simple regular languages. Finally, we show that sets of squares recognized by DFAs can be as sparse as any recursively enumerable set.

1. INTRODUCTION

Two-dimensional languages, or *picture languages,* are an interesting generalization of the standard languages of computer science. Rather than one-dimensional strings, we consider two-dimensional arrays of symbols over a finite alphabet. These arrays can then be accepted or rejected by various types of automata, and this gives rise to different language classes. Such classes may be of interest as formal models of image recognition, or simply as mathematical objects in their own right.

Much of this work has focused on two- or more-dimensional generalizations of regular languages. We can define the regular languages as those recognized by finite-state automata that can move in one direction or both directions on the input, and which are deterministic or non-deterministic. We can also consider finite complement languages, in which some finite set of subwords is forbidden, and then project onto a smaller alphabet with some alphabetic homomorphism. In one dimension, these are all equivalent in their computational power.

In two dimensions, we can consider *4-way finite-state automata*, which at each step can read a symbol of the array, change their internal state, and move up, down, left or right to a neighboring symbol. These can be deterministic or non-

[1]Research supported by NSF Grant CCR 97-33101

deterministic, and DFAs and NFAs of this kind were introduced by Blum and Hewitt [1]. Similarly, we can forbid a finite number of subblocks and then project onto a smaller alphabet, obtaining a class of picture languages which we call *homomorphisms of local lattice languages,* or h(LLL)s [8]. (These are also called the *recognizable* languages [3] or the languages recognizable by *non-deterministic on-line tesselation acceptors* [5].) While DFAs, NFAs and h(LLL)s are equivalent in one dimension, in two or more they become distinct:

$$\text{DFA} \subset \text{NFA} \subset \text{h(LLL)}$$

where these inclusions are strict. Reviews of these classes are given in [8, 4, 7, 11], and a bibliography of papers in the subject is maintained by Borchert at [2].

A fair amount is known about the closure properties of these classes as well. The DFA, NFA, and h(LLL) languages are all closed under intersection and union using straightforward constructions.

The situation for complement is somewhat more complicated. DFAs are closed under complement by an argument of Sipser [12] which allows us to remove the danger that a DFA might loop forever and never halt. We construct a new DFA that starts in the final halt state, which we can assume without loss of generality is in the lower right-hand corner. Then this DFA does a depth-first search backwards, attempting to reach the initial state of the original DFA. (We use a similar construction in Section 4 below.) This gives a loop-free DFA which accepts if and only if the original DFA accepts, and since a loop-free DFA always halts, we can then switch accepting and rejecting states to recognize the complement of the original language.

It is also known that h(LLL)s are not closed under complement. However, up to now it has been an open question whether NFAs are closed under complement or not [4, 8].

In this paper, we resolve this question and show that NFAs are not closed under complement. In addition, we show that NFAs are more powerful than DFAs, even for rectangles of a single symbol. We note that recognition of rectangles by NFAs, in the context of "recognizable functions," is studied in [6], where some upper bounds are shown.

We then consider the question of what sets of rectangles can be recognized by an NFA or DFA from the outside, i.e. by an automaton which is not allowed to enter the rectangle but can roam the plane outside it. We show that any such set corresponds to a simple regular language, and therefore that such an automaton can be simulated by a DFA moving along the rectangle's boundary.

Finally, we show that DFAs can recognize squares whose sizes belong to sets which are as sparse as any recursively enumerable set. This resolves an open question raised in [8] about how sparse these sets can be.

2. NFAS AND DFAS OVER A ONE-SYMBOL ALPHABET

In this section, we give an example of a rectangle set S that can be recognized by a two-dimensional NFA, but not by any DFA. Moreover, we show that the complement of S is not recognized by any NFA. This proves that, unlike in the deterministic case, the class of picture languages recognizable by NFAs is not closed under complementation, even for pictures over a one-symbol alphabet.

Our main tool in proving negative results is to interpret two-dimensional automata as two-way one-dimensional automata by fixing the height of the rectangles and letting the width vary. Our variant of two-way finite automata can detect when they are reading the first or the last letter of the input and can use different transition rules there. They may move left or right, or remain stationary. They cannot move beyond the ends of the input. A word is accepted iff a final state can be reached.

LEMMA 2.1. *Let $S \subseteq \mathbb{N}^2$ be the set of rectangles recognized by a k-state NFA A. Then for every height h there exists a two-way NFA B with kh states recognizing the language*

$$\{1^w \mid (w, h) \in S\}$$

of corresponding widths. Moreover, if A is deterministic then B is deterministic.

Proof. The states of B are pairs (i, s) where i is an integer, $1 \leq i \leq h$, representing the current vertical position of A in the rectangle, and s is the current state of A. The position of B represents the horizontal position of A inside the rectangle. It is easy to interpret A's transition rules as B's transition rules, so that B will simulate A step-by-step. If A is deterministic then B is also. ■

The following "pumping lemma" allows us to prove that certain languages cannot be accepted by any two-way NFA or DFA with a given number of states:

LEMMA 2.2. *Let A be a two-way NFA with k states over the single letter alphabet $\{1\}$, accepting a language $L \subseteq 1^*$. Then, for every $n > k + 2$,*

$$1^n \in L \Longrightarrow 1^{n+k!} \in L$$

Moreover, if A is deterministic then the implication above becomes an equivalence "$\Longleftrightarrow$".

Proof. Let $n > k+2$ and consider an accepting computation C of input 1^n. Let us divide C into segments between consecutive visits of the NFA at the endpoints of the input. The head has to move through all intermediate positions during any segment S of C from one end to the other. There are $n - 2 \geq k + 1$ intermediate positions. Let $s_1, s_2, \ldots, s_{n-2}$ be the states of A when it enters the intermediate positions $1, 2, \ldots, n-2$, respectively, for the first time within segment S. It follows from the pigeonhole principle that two of the states $s_1, s_2, \ldots, s_{k+1}$ must be the same, say $s_i = s_{i+t}$. The computation between positions i and $i + t$ can then be repeated arbitrarily many times, taking the NFA into position $i + jt$ for any $j \geq 0$, and remaining in state s_i.

Because t divides $k!$ this means that the input $1^{n+k!}$ is accepted by a computation that is identical to C except that in any segment of C from one end to the other a loop of length t is repeated $k!/t$ times.

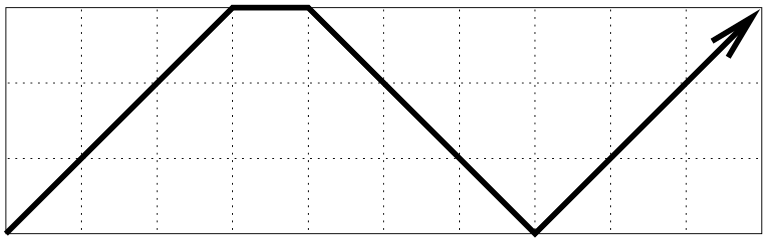

FIG. 1. An accepting computation for the 10×3 rectangle.

If A is deterministic then each input has a unique computation C. If $n > k+2$ and 1^n is not accepted (either A halts in non-accepting state or loops forever) then the same construction as above yields a non-accepting computation (halting or looping) for input $1^{n+k!}$. In other words, in the deterministic case we have $1^n \in L \iff 1^{n+k!} \in L$. ■

This gives us the immediate corollary:

Corollary 2.1. *If L is a finite language recognized by a two-way NFA with k states, then its longest word has length at most $k+2$.*

Now we are ready to prove the main results of this section. Consider the following set of rectangles:

$$\begin{aligned} S &= \{(w,h) \,|\, w = ih + j(h+1) \text{ for some non-negative integers } i \text{ and } j\ \} \\ &= \{(w,h) \,|\, w = ih + j \text{ for some } 0 \leq j \leq i\ \}. \end{aligned}$$

For any given height h the set of allowed widths w is the union of contiguous segments

$$ih, ih+1, \ldots, ih+i$$

for all $i = 0, 1, \ldots$. It is easy to see that the largest width w that is not allowed is $h^2 - h - 1$. (It is the only integer between the segments for $i = h-2$ and $i = h-1$. For larger values of i the consecutive segments overlap.)

The set S can be easily recognized by a billiard-ball-like NFA that sets out at a 45° angle from the lower left corner of the rectangle. When the ball hits the upper or lower edge of the rectangle it either bounces back immediately, or moves one cell to the right along the edge before bouncing, as in Figure 1. The rectangle is accepted if the ball is in either corner of the rectangle when it reaches the right edge.

However, the complement $\overline{S}$ is not accepted by any NFA. Notice that $\overline{S}$ contains a finite number of rectangles of any given height h, the largest of which has width $w = h^2 - h - 1$. Assume that $\overline{S}$ is accepted by an NFA with k states. According to Lemma 2.1 there exists a two-way NFA B with kh states that accepts the finite language

$$L = \{1^w \,|\, (w,h) \in \overline{S}\}.$$

If we choose the height h such that $h^2 - h - 1 > kh + 2$, then the longest word in L has length greater than the number of states in B plus two. This contradicts the corollary to Lemma 2.2, so we have proved

THEOREM 2.1. *The family of rectangular sets recognizable by NFAs is not closed under complementation.*

Because the family of rectangular sets recognizable by DFA is closed under complementation [12] we have

COROLLARY 2.2. *There are rectangular sets recognizable by NFAs that are not recognized by any DFA.*

An example of such a set is our set S above. We can show it directly, without the help of closure properties, using the deterministic variants of Lemmmas 2.1 and 2.2.

3. RECOGNIZING RECTANGLES FROM OUTSIDE

Let us consider next the "dual" problem of recognizing rectangles using a finite automaton that operates outside the rectangle and is not allowed to penetrate inside. Not surprisingly, such devices turn out to be less powerful than automata that operate inside rectangles. We prove that if $S \subseteq \mathbb{N}^2$ is the set of rectangles recognized by an NFA from outside, then the language

$$L_S = \{0^i 1^j \mid (i, j) \in S\}$$

is regular. In other words, the same set of rectangles can be recognized by a DFA that moves east and south along the edges of the rectangle from the upper left corner to the lower right corner. A similar result was shown for DFAs which are allowed to make excursions into the plane outside the rectangle by Milgram [9].

LEMMA 3.1. *Let $S \subseteq \mathbb{N}^2$ be recognized from the outside by an NFA A. Then there exist positive integers t_H and p_H, called the horizontal transient and the period, respectively, such that for every $w > t_H$ and every h*

$$(w, h) \in S \Longrightarrow (w + p_H, h) \in S$$

where the numbers t_H and p_H are independent of the height h.

Proof. We generalize an argument from [8] for rectangles of height 1. Consider first computation paths P by A on unmarked plane that start in state q at position $(0, 0)$ and end in state r on the same row $y = 0$. Assume that P is entirely above that row, i.e. all intermediate positions (x, y) of P have $y > 0$. Let us prove the following fact: there must exist positive integers t and p such that, if P ends in position $(x, 0)$ with $x > t$ then there is a computation path P' from the same initial position $(0, 0)$ and state q, into position $(x + p, 0)$ and state r, such that also P' is entirely above the row $y = 0$. In other words, the possible horizontal positions where A returns to the initial row $y = 0$ in state r are eventually periodic.

To prove this, notice that sequences I of instructions executed by A during possible P's form a context-free language, accepted by a pushdown automaton that uses its stack to count its vertical position and its finite state control to simulate the states of A. Recall that a Parikh vector of a word is the integer vector whose elements are the counts of different letters in the word. It is well known that in the case of a context-free language the Parikh vectors of the words in the language form a semi-linear set, that is, a finite union of linear sets $\{\overline{u} + \sum_{i=0}^{k} a_i \overline{v}_i \mid a_i = 0, 1, \ldots\}$ for integer vectors $\overline{u}$ and $\overline{v}_i$, $i = 0, 1, \ldots, k$.

Any linear transformation keeps linear sets linear; therefore, the possible values of $n_r(I) - n_l(I)$ form a semilinear set of integers, where $n_r(I)$ and $n_l(I)$ indicate the numbers of right- and left-moving instructions in the word I, respectively. Because $n_r(I) - n_l(I)$ is the total horizontal motion caused by executing I, we conclude that possible horizontal positions at the end of P form a semi-linear set $\cup_j \{t_j + ip_j \mid i = 0, 1, \ldots\}$. By choosing t to be the maximum of all the t_j's and p as the lowest common multiple of the positive p_j's, we obtain numbers that satisfy the following claim: if $x > t$ is the horizontal position at the end of some P, then $x = t_j + ip_j$ for some numbers $i \geq 1$ and j with $p_j > 0$. Because p_j divides p we see that $x + p$ is in the same linear set $\{t_j + ip_j \mid i = 0, 1, \ldots\}$ as x is.

As proved above, the transients t and the periods p exist for all start states q and end states r. We can find a common transient and period for all start and end states by taking the lowest common multiple of the periods p and the largest of the transients t.

Now we are ready to proceed with the proof of the Lemma. Let us divide the plane outside a rectangle into four parts as indicated in Figure 2. Parts L and R are semi-infinite strips to the left and to the right of the rectangle. Consider an accepting computation C for a $w \times h$ rectangle and let us divide C into segments at the transitions between the four parts. More precisely, a segment will be a part of C that is entirely inside one of the four regions, and the automaton touches the border between two regions only at the beginning and end of each segment.

If the width w of the rectangle is increased, all segments of C can be executed unchanged except segments that take the automaton from L to R or from R to L through areas U or D.

Let us study a segment that takes the automaton from L to R through U. Let us divide it into subsegments that start and end on the same row as the beginning and end of the entire segment, so that inside the subsegments all positions are strictly above that row. See the illustration in Figure 3. Notice that the subsegments start and end in positions that touch the upper edge of the rectangle. The only exceptions are the start and end positions of the first and last subsegment, which lie above L and R respectively.

If any of the subsegments moves the automaton more than t cells to the right where t is the number from the first part of the proof, then according to the first part that subsegment can be replaced by another one that moves the automaton $k \cdot p$ additional cells to the right, for any $k = 1, 2, \ldots$. This allows us to 'pump' this segment of the computation farther to the right.

If, on the other hand, every subsegment moves A either to the left or at most t cells to the right, we can still pump the segment farther right as follows. Let s be the number of states in A. If the width w of the rectangle is at least $(s + 1)t$ then

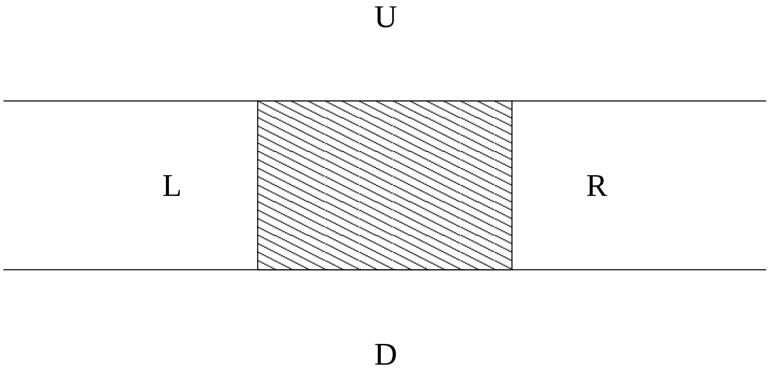

FIG. 2. Four parts of the plane.

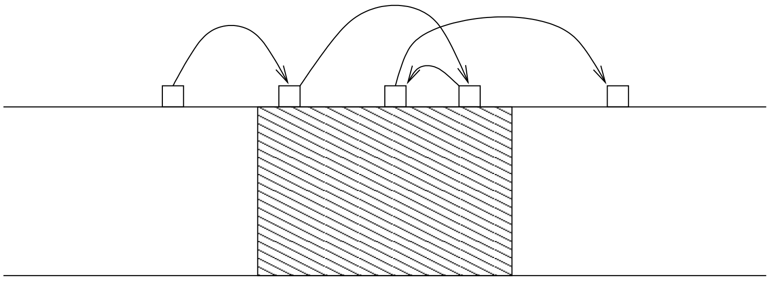

FIG. 3. A sample segment under consideration.

the automaton must touch the upper edge of the rectangle at an increasing series of times at positions $x_1 < x_2 < x_3 < \cdots < x_{s+1}$, all satisfying $x_{i+1} - x_i \leq t$. It follows from the pigeonhole principle that the automaton must be in the same state when it touches the rectangle in two different positions x_i and x_j for some $i < j$. The computation path between positions x_i and x_j can be repeated arbitrarily many times, leading to an additional movement by $k \cdot (x_j - x_i)$ cells to the right, for any chosen $k = 1, 2, \ldots$. Because $x_j - x_i \leq st$ we conclude that an additional movement to the right by any multiple of $(st)!$ cells is possible.

From the two cases above it follows that if the width of the rectangle is at least $(s+1)t$ then path P can be modified to cross an additional distance of the lowest common multiple of $(st)!$ and p. Therefore, the width of the rectangle can be increased by $\mathrm{lcm}(p, (st)!)$.

A similar argument can be made for segments that cross from R to L, or between R and L through the lower half D. By taking the maximum of the transient lengths and the lowest common multiple of the periods we obtain numbers t_H and p_H that satisfy the lemma. The numbers are independent of the height of the rectangle. ∎

The previous lemma has, naturally, a vertical counterpart:

LEMMA 3.2. *Let $S \subseteq \mathbb{N}^2$ be recognized from outside by an NFA A. Then there exist positive integers t_V and p_V such that for every $h > t_V$ and every w*

$$(w, h) \in S \Longrightarrow (w, h + p_V) \in S$$

where the numbers t_V and p_V are independent of the width w.

Finally, we use the following technical result:

LEMMA 3.3. *Assume that a language $L \subseteq 0^*1^*$ satisfies the following monotonicity conditions: there exist positive integers n and m such that*

$$\begin{array}{lll} 0^i1^j \in L, i \geq n & \Longrightarrow & (\forall x \geq i)\, 0^x1^j \in L, \text{ and} \\ 0^i1^j \in L, j \geq m & \Longrightarrow & (\forall y \geq j)\, 0^i1^y \in L. \end{array}$$

Then L is regular.

Proof. For every fixed k the language

$$A_k = L \cap 0^k1^*$$

is regular: if for some $j \geq m$ we have $0^k1^j \in L$, then A_k is the union of $0^k1^j1^*$ and a finite number of words that are shorter than 0^k1^j. If no such j exists then A_k is finite and therefore regular. Analogously, languages

$$B_k = L \cap 0^*1^k$$

are regular.

If L contains some word 0^i1^j with $i \geq n$ and $j \geq m$ then

$$\begin{array}{lll} L & = & A_0 \cup A_1 \cup \cdots \cup A_{i-1} \\ & & \cup B_0 \cup B_1 \cup \cdots \cup B_{j-1} \\ & & \cup 0^i0^*1^j1^* \end{array}$$

is regular since it is a finite union of regular languages. If no such word 0^i1^j is in L then

$$\begin{array}{lll} L & = & A_0 \cup A_1 \cup \cdots \cup A_{n-1} \\ & & \cup B_0 \cup B_1 \cup \cdots \cup B_{m-1} \end{array}$$

is a union of a finite number of sets of the form A_k and B_k, and is therefore regular. ∎

Now we have all the necessary tools to prove the main result of the section:

THEOREM 3.1. *Suppose an NFA recognizes from outside a set $S \subseteq \mathbb{N}^2$ of rectangles. Then the language*

$$L_S = \{0^i1^j \mid (i,j) \in S\}$$

is regular.

Proof. Let t_H and p_H be the horizontal transient and the period from Lemma 3.1, and let t_V and p_V be their vertical counterparts from Lemma 3.2. For every $a = 0, 1, \ldots, p_H - 1$ and $b = 0, 1, \ldots, p_V - 1$ let us define

$$L_{a,b} = L_S \cap \{0^i1^j \mid i \equiv a \pmod{p_H}, j \equiv b \pmod{p_V}\}.$$

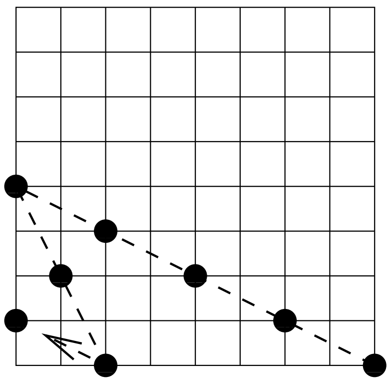

FIG. 4. Using knight's moves to recognize $2^n \times 2^n$ squares.

Because L_S is the union of languages $L_{a,b}$ it is enough to prove that every $L_{a,b}$ is regular.

But it follows from Lemmas 3.1 and 3.2 that the language

$$L'_{a,b} = \{0^{(i-a)/p_H} 1^{(j-b)/p_V} \mid 0^i 1^j \in L_{a,b}\}$$

satisfies the monotonicity condition of Lemma 3.3 and is therefore regular. If h is the homomorphism that replaces 0 with p_H 0's and 1 with p_V 1's then

$$L_{a,b} = 0^a h(L'_{a,b}) 1^b.$$

Thus $L_{a,b}$ is regular. ■

4. SQUARE RECOGNITION

A DFA can easily check if a given rectangle is a square by moving diagonally through the rectangle. Since the class of sets recognized by DFAs or NFAs are closed under intersection, the set of squares recognized by a DFA or NFA is recognizable by an automaton of the same type. A natural question to ask is what sets of squares can be recognized by NFA or DFA. We say that a set $S \subset \mathbb{N}$ is *square-recognizable* by an NFA (DFA) if the set of squares $\{(n,n) \mid n \in S\}$ is recognizable by an NFA (DFA).

Example 1: [8] The set $\{2^n \mid n \in \mathbb{N}\}$ is square-recognized by a DFA that uses knight's moves to divide the distance to a corner by two, as in Figure 4.

Example 2: A slightly more complex construction shows that the set $\{2^{2^n} \mid n \in \mathbb{N}\}$ can be square-recognized. In this construction a signal of rational speed 3/4 is repeatedly used to move the automaton from position 4^n into position 3^n, in the same style as the knight moves was used in the previous example. That is, we multiply the automaton's position on an edge by 3/4 until the result is no longer an integer. Then signals of speed 2/3 are iterated to move the automaton from position 3^n into position 2^n. This process accomplishes the following changes in the automaton's position:

$$2^{2^n} = 4^{2^{n-1}} \longrightarrow 3^{2^{n-1}} \longrightarrow 2^{2^{n-1}}.$$

The process is iterated until position $2 = 2^{2^0}$ is reached, in which case the square is accepted. If it ends at position 1 instead, we reject. Note that we can check the position mod 2, 3, or 4 at any time by using a finite number of states. At the very beginning we can verify that the square size is a power of 2, using the DFA from our first example.

We see from these examples that rather "sparse" sets of squares can be recognized, even by deterministic finite automata. In what follows we prove that these sets can be made as sparse as any recursive (or recursively enumerable) set. This result is optimal because any square-recognized set has to be recursive.

THEOREM 4.1. *Let $\{a_1, a_2, \ldots\}$ be any recursively enumerable set of positive integers. There exists a 2D DFA that square-recognizes a set $\{b_1, b_2, \ldots\}$ such that $b_i > a_i$ for all $i = 1, 2, \ldots$*

Proof. We exploit classical results by Minsky [10] and a very natural correspondence between automata inside rectangles and 2-counter machines without an input tape. A 2-counter machine without an input tape consists of a deterministic finite state automaton and two infinite registers, called counters. The counters store one non-negative integer each. The machine can detect when either counter is zero. It changes its state according to a deterministic transition rule. The new state only depends on the old state and the zero vs. non-zero status of the two counters. The transition rule also specifies how to change the counters: They may be incremented, decremented (only if they are non-zero) or kept unchanged. The automaton accepts iff it reaches a specified accepting state. We define the set recognized by such a device as the set of integers k such that the machine enters the accepting state when started with counter values 0 and k.

It follows from Minsky's construction in [10] that for every recursively enumerable set X of positive integers there exists a 2-counter machine A that accepts the set

$$Y = \{2^n \mid n \in X\}.$$

Moreover, we may assume that the 2-counter machine A never loops, i.e., never returns back to the same state and counter values it had before. This assumption means no loss of generality because we can start Minsky's construction from a Turing machine that does not loop. Such Turing machine exists for every recursively enumerable set. (It may, for example, count on the tape the number of instructions it executes.)

Notice that a 2-counter machine can be interpreted as a 2D DFA that operates on the (infinite) positive quadrant $\{(x, y) \mid x, y \geq 0\}$ of the plane and has the same finite states as the 2-counter machine. The values of the counters correspond to the coordinates of the position of the automaton. Increments and decrements of the counters correspond to movements of the DFA on the plane. Counter values zero are the positive axeses $(x, 0)$ and $(0, y)$, which can be detected by the DFA.

An accepting computation of a 2-counter machine can therefore be simulated by a 2D DFA inside a sufficiently large square. The square has to be at least as large as the largest counter value used during the accepting computation. Using this observation we prove that if Y is any set of positive integers accepted by a

2-counter machine A that does not loop then some 2D DFA square-recognizes the set

$$Z = \{k \in \mathbb{N} \mid \quad k \text{ is the largest counter value during the accepting computation by } A \text{ of some } i \in Y \}.$$

It is very easy to construct a non-deterministic 2D automaton that square-recognizes set Z: The NFA moves along the lower edge of the square and uses the non-determinism to guess the starting position of the 2-counter machine A. Then it executes A until A accepts, halts otherwise or tries to move outside the square. Moreover, the NFA memorizes in the finite control unit if it ever touched the right or upper edge of the square, indicating that a counter value equal to the size of the square was used. The square is accepted if and only if A accepts after touching the right or the upper edge of the square. Note that the forward simulation part of this process is deterministic. Non-determinism is only needed to find the correct input to the 2-counter machine.

To accept set Z deterministically we use the determinism of the 2-counter machine to depth-first-search the predecessor tree of a given configuration C to see if it is on a computation path for some valid input. This is done by trying all possible predecessors of a configuration in a predetermined order, and recursively processing all these predecessors. When no predecessors exist the DFA can backtrack using the determinism of the 2-counter machine, and continue with the next branch of the predecessor tree. Note that the predecessor tree – restricted inside a square – is finite and does not contain any loops so the depth-first-search process will eventually terminate. It finds a valid starting configuration that leads to the given configuration C if it exists. If no valid input configuration exists the process may identify an incorrect starting configuration if the depth-first-search backtracked beyond C. In any case, if a potential start configuration is found, a forward simulation of the 2-counter machine is done to see if it accepts after touching the right or the upper edge of the square.

The depth-first-search process described above is done on all configurations C that are close to the upper left or the lower right corners of the input square. If the two counter machine A has s states then any accepting computation with maximum counter value k must enter some cell within distance s of one of the corners: otherwise counter value k is present in a loop that takes the automaton away from the axeses, and the computation cannot be accepting.

There are a finite number of configurations C within distance s from the corners, so the 2D DFA can easily try all of them one after the other.

Now we are ready to conclude the proof. If $X = \{a_1, a_2, \ldots\}$ is any r.e. set of positive integers then there exists a 2-counter machine A that does not loop and accepts $Y = \{2^{a_1}, 2^{a_2}, \ldots\}$. The construction above provides a 2D DFA that square accepts the set $Z = \{b_1, b_2, \ldots\}$ where b_i is the maximum counter value during the accepting computation of 2^{a_i} by A. Clearly $b_i \geq 2^{a_i} > a_i$. ∎

Another way of stating the previous theorem is

Corollary 4.1. *Let $f : \mathbb{N} \longrightarrow \mathbb{N}$ be any computable function. Then some 2D DFA square accepts the range of some function $g : \mathbb{N} \longrightarrow \mathbb{N}$ that satisfies $g(n) \geq f(n)$ for all $n \in \mathbb{N}$.*

It is interesting to note that there seem to be no tools whatsoever, at the present time, to prove that a set S cannot be square-recognized by a DFA. To inspire the creation of such tools, we make the following conjecture:

Conjecture 1. No DFA or NFA in two dimensions can recognize the set of squares of prime size. ■

In contrast, the reader can easily show that there is a two-dimensional DFA that recognizes squares of size 11^p for p prime. This works by first moving p from the 11-register to the 2-register, that is, by converting 11^p to 2^p as in Example 2 above. It then copies p into to the 3-register, and uses the 5- and 7-registers to maintain a number m by which it divides p, with a loop that passes m back and forth between the 5- and 7-registers while decrementing the 3-register. After each division, it increments m and recopies p from the 2-register to the 3-register. We halt if m exceeds $\lfloor p/4 \rfloor$, which we can check with a similar loop; this suffices for all $p > 9$. Then since $2^p 3^p 7^{p/4} \leq 11^p$ all this can take place inside a square of size 11^p.

In addition, in [8] DFAs are given that recognize cubes of prime size, and tesseracts of perfect size, in three and four dimensions respectively. These use a billiard-ball-like computation to check whether two integers are mutually prime; thus they simulate counter machines which branch both when the counter is zero and when it reaches its maximum, i.e. the size of the cube.

We also conjecture the following:

Conjecture 2. NFAs are more powerful than DFAs on squares of one symbol. ■

Since we have proved that NFAs are more powerful than DFAs on rectangles of one symbol, it would be rather strange if they were equivalent on squares.

Acknowledgments. JK thanks the Santa Fe Institute for an enjoyable visit where this work began. We also thank Bernd Borchert and Juraj Hromkovič for helpful conversations.

REFERENCES

1. M. Blum and C. Hewitt (1967), Automata on a 2-dimensional tape. *8th IEEE Symp. on Switching and Automata Theory* 155–160.
2. B. Borchert, `http://math.uni-heidelberg.de/logic/bb/2dpapers.html`
3. D. Giammarresi and A. Restivo (1992) Recognizable picture languages. *Int. J. of Pattern Recognition and Artificial Intelligence* **6(2-3)** 241-256.
4. D. Giammarresi and A. Restivo (1996) Two-dimensional languages. In G. Rosenberg and A. Salomaa, Eds., *Handbook of Formal Languages,* Volume III, pp. 215–267. Springer-Verlag.
5. K. Inoue and A. Nakamura (1977) Some properties of two-dimensional on-line tesselation acceptors. *Information Sciences* **13** 95–121.
6. K. Inoue and A. Nakamura (1979) Two-dimensional finite automata and unacceptable functions. *Int. J. Comput. Math. A* **7** 207–213.
7. K. Inoue and I. Takanami (1991) A survey of two-dimensional automata theory. *Information Sciences* **55** 99–121.
8. K. Lindgren, C. Moore and M.G. Nordahl (1998) Complexity of two-dimensional patterns. *Journal of Statistical Physics* **91** 909–951.

9. D.L. Milgram (1976) A region crossing problem for array-bounded automata. *Information and Control* **31** 147–152.
10. M. Minsky (1967) *Computation: Finite and Infinite Machines.* Prentice-Hall.
11. A. Rosenfeld (1979) *Picture Languages: Formal Models for Picture Recognition.* Academic Press.
12. M. Sipser (1980) Halting space-bounded computations. *Theoretical Computer Science* **10** 335-338.