Submitted to:
Information Processing Letters

Recognizing Majority on a One-way Mesh

Carla Savage

Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27695

CCSP-TR-87/18

October 1987

This work was supported in part by the Center for Communications and Signal Processing, North Carolina State University

Abstract

The two-dimensional majority language is the set of all two-dimensional arrays with 0/1 entries in which there are more 1's than 0's. We show that this language can be recognized by a two-dimensional bounded cellular automaton using only one-way communication between neighboring cells. The recognition can occur within diameter time.

1. Introduction

An open question in the study of arrays of finite state machines is whether or not there is any loss of computing power when communication between neighbors is restricted to be one-way rather than two-way. This problem has been addressed by several researchers, including [Dye80], [UMS82], [CY85], and [CIV86], and is likely to be hard in view of results obtained in [CIV86].

In this paper we consider a relevant problem which was posed in [Dye80], that of recognizing the two-dimensional majority language (the set of all two-dimensional 0/1 arrays containing more 1's than 0's) using only a one-way bounded cellular automaton. It was known that this language could be recognized (in diameter time) by a two-way bounded cellular automaton [Kos74].

In the remainder of this section, we give the necessary definitions. Section 2 describes the algorithm and analyzes its time.

A *one-way two-dimensional bounded acceptor*, M, consists of an array $M[1..n,1..m]$ of identical finite state machines in which each machine $M[i,j]$ has

- finite state set Q
- boundary state $\# \notin Q$
- input alphabet $I = (Q \cup \{\#\}) \times (Q \cup \{\#\})$
- transition function $\delta: I \times Q \rightarrow Q$

- accepting state set $Q_A \subset Q$

A *configuration* C of M is an array $C[0..n,0..m]$ of elements of $Q \cup \{\#\}$ such that

if $i = 0$ or $j = 0$

then $C[i,j] = \{\#\}$

else $C[i,j] \in Q$

M operates in a sequence of discrete time units. Given an input configuration C_0, M assumes configuration C_t at time t where for $t > 0$,

$$C_t[i,j] = \delta((C_{t-1}(i,j-1),C_{t-1}(i-1,j)),C_{t-1}(i,j)).$$

M *accepts* input configuration C_0 if $M[n,m]$ ever enters an accepting state, that is if there exists a $t \geq 0$ with $C_t[n,m] \in Q_A$.

We note that this definition could be extended to one-way k-dimensional bounded cellular acceptors and corresponds to the "k-way k-dimensional bounded cellular acceptors" defined in [Dye80].

2. Recognizing Majority

The majority language is the set of all rectangular 0/1 matrices containing more 1's than 0's. The problem is to construct a one-way, two-dimensional bounded acceptor $M[1..n,1..m]$ such that for any initial configuration $C[0..n,0..m]$ with

$C[i,j] = \#$, if $i = 0$ or $j = 0$

otherwise, $C[i,j] = 0$ or 1,"

M enters an accepting state if and only if $C[1..n,1..m]$ is an element of the majority language.

To see how this can be done, let

$sub(i,j)$ = (number of 0's) − (number of 1's) in $C[1..i,1..j]$.

$row(i,j)$ = (number of 0's) − (number of 1's) in subrow $C[i,1..j]$, that is, in entries 1 through j of row i of C.

$inc(i,j) = 1$, if $C[i,j] = 0$, otherwise -1.

Then the values for sub, row, and inc satisfy the following relationships.

if $i = 1$ and $j = 1$ then {upper left corner}

$sub(i,j) = inc(i,j)$

$$row(i,j) = inc(i,j)$$

otherwise, if $j = 1$ then {left column}

$$sub(i,j) = sub(i-1,j) + inc(i,j)$$

$$row(i,j) = inc(i,j)$$

otherwise, if $i = 1$ then {top row}

$$sub(i,j) = row(i,j-1) + inc(i,j)$$

$$row(i,j) = row(i,j-1) + inc(i,j)$$

otherwise,

$$sub(i,j) = sub(i-1,j) + row(i,j-1) + inc(i,j)$$

$$row(i,j) = row(i,j-1) + inc(i,j)$$

Thus, given an initial configuration $C[1..m,1..n]$ of 0's and 1's, in a one-way, bounded array of machines *which were not restricted to be finite state* the corner cell (1,1) could, at time 1, detect itself (from the symbols # to the left and above) and compute $sub(1,1)$ and $row(1,1)$. Then for $i+j > 2$, cell (i,j) can compute $sub(i,j)$ and $row(i,j)$ at time $i+j-1$ from the values computed by cells $(i,j-1)$ and $(i-1,j)$ at time $i+j-2$. (See Figures 1 and 2.)

However, since machines must be finite state, values can be computed and stored by the cells only a finite number of bits at a time. That is, each finite state machine in the array M must be capable of accepting, one bit at a time,

some representation of the values *sub* and *row* in its two neighbors (left and above), and producing, in response, the next bit of the representation of its own *sub* and *row* values. This can be done using a combination of two basic ideas. First, we use two's complement representation for negative numbers and secondly, we use exactly $i+j$ bits to represent the integers $sub(i,j)$ and $row(i,j)$ to circumvent problems of overflow. Characters '<' and '>' are used to precede the first bit and to follow the last bit, respectively. Figure 3 illustrates these representations for the *sub* and *row* values in Figure 2.

We now review two's complement representation (see for example [Sta87]). Let x be an integer satisfying

$$|x| < 2^{k-1}$$

so that $|x|$ can be represented by $k-1$ bits. Let $f_k(x)$ be the two's complement representation of x, defined by

$$f_k(x) = x \text{ if } x \geq 0$$

$$\text{otherwise, } f_k(x) = 2^k - |x|$$

The following observations can be made for $|x| < 2^{k-1}$.

(i) If $x < 0$, then $f_k(x)$ can be formed by scanning the binary representation of $|x|$, from low order bit to high order bit: leave all bits unchanged up to and including the first 1; then complement all

remaining bits.

(ii) The highest order bit (k-th bit) of $f_k(x)$ is 0 if x is positive and 1 if x is negative.

(iii) If $f_k(x) = x_{k-1}x_{k-2} \cdots x_0$ then $f_{k+1}(x) = x_{k-1}x_{k-1}x_{k-2} \cdots x_0$.

(iv) $f_k(x)$ is always positive.

For $x \geq 0$, let $[x]_k$ denote the binary number represented by the k lowest order bits of the binary representation of x. Then the following can be shown.

Theorem. If $|x+y| < 2^{k-1}$ then

$$[f_k(x+y)]_k = [f_k(x)+f_k(y)]_k.$$

Proof. By cases: If both x and y are nonnegative, the theorem is clear. If x and y are both negative, then by definition of f_k, we have

$$f_k(x)+f_k(y) = 2^k - |x| + 2^k - |y| = 2^{k+1} - |x+y| = f_{k+1}(x+y)$$

If $x \geq 0$, $y < 0$, and $x+y \geq 0$ then

$$f_k(x)+f_k(y) = x+2^k-(-y) = x+y+2^k.$$

But since $|x+y| < 2^{k-1}$.

$$[x+y+2^k]_k = x+y = f_k(x+y).$$

If $x \geq 0, y < 0$, and $x+y < 0$ then

$$f_k(x)+f_k(y) = x+2^k-(-y) = 2^k--(x+y) = f_k(x+y).$$

Cases for x negative and y positive are symmetric.

□

As a consequence of the theorem, as long as $|x+y| < 2^{k-1}$, the k bits of $f_k(x+y)$ can be obtained from the lowest order k bits of the binary sum, $f_k(x)+f_k(y)$. Further, we can obtain the $k+1$ bits of $f_{k+1}(x+y)$ from the k bits of $f_k(x+y)$ simply by repeating the highest order bit.

Finally, it can be shown as in the theorem that, given $f_k(x)$ and $f_k(y)$ one bit at a time (lowest to highest order), it is straightforward to compute, one bit at a time all of the following, provided the argument of f_{k+1} is smaller than 2^{k-1}:

$$f_{k+1}(x+1)$$

$$f_{k+1}(x-1)$$

$$f_{k+1}(x+y+1)$$

$$f_{k+1}(x+y-1)$$

We apply these ideas to recognizing majority with a one-way two-dimensional bounded cellular array. For $i > 1$ and $j > 1$, cell (i,j) will receive, one bit at a time, the $(i+j-1)$-bit two's complement representation of the *sub* and *row* values from cells $(i,j-1)$ and $(i-1,j)$. Cell (i,j) will compute, a bit at a time, the $(i+j)$-bit two's complement representation of $sub(i,j)$ and $row(i,j)$. The input and output strings for cell (i,j) are preceded by a '<' and followed by a '>' (see Figure 4).

Since ij is an upper bound on $|\, sub(i,j)\,|$ and $|\, row(i,j)\,|$, $i+j$ bits will suffice for the two's complement representation of $sub(i,j)$ and $row(i,j)$. Cell (i,j) enters an accepting state if and only if the last bit of $sub(i,j)$ is a one. (This condition is detected when cell (i,j) receives the pair $(>,>)$ from its two neighbors.

To get the computation started, initially, cell (1,1) is in state 0 or 1. To determine its state at time $t = 1$, cell (1,1) receives the states (#,#) from the left and above and therefore knows it is the corner cell. It enters the sequence of states shown in Figure 5.

If cell (i,j) is going to enter an accepting state, it does so by the time it outputs the endmarker '>'.

Let $T(n,m)$ be the time required by an n by m cellular array to recognize majority using this algorithm. The time satisfies the following equations.

If $i = j = 1$ then $T(i,j) = 4$

otherwise, if $i \neq 1$ then $T(i,j) = T(i-1,j)+2$

otherwise $T(i,j) = T(i,j-1)+2$.

Thus, $T(n,m) = 2(n+m)$, which is diameter time, as defined in [Dye80].

References

[CIV86] Chang, J.H., O.H. Ibarra, and A. Vergis, "On the power of one-way communication," *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science,* Toronto, October 1986.

[CY85] Culik, K. and S. Yu, "Translation of systolic algorithms between systems of different topology," *Proceedings, International Conference on Parallel Processing,* 1985.

[Dye80] Dyer, C., "One-way bounded cellular automata", *Information and Control 44,* 1980.

[Sta 87] Stallings, W., *Computer Organization and Architecture,* MacMillan 1987.

[Kos74] Kosaraju, S.R., "On some open problems in the theory of cellular automata," *IEEE Transactions on Computers,* Vol c-23, No. 6, June 1974.

[UMS82] Umeo, H., M. Kenichi, and K. Sugata, "Deterministic one-way simulation of two-way real time cellular automata and its related problems," *Information Processing Letters,* Vol. 14, No. 4, June 1982.

	#	#	#	#
#	0	1	0	1
#	1	1	0	0
#	1	1	1	0
#	1	0	1	0

Figure 1. Initial Configuration

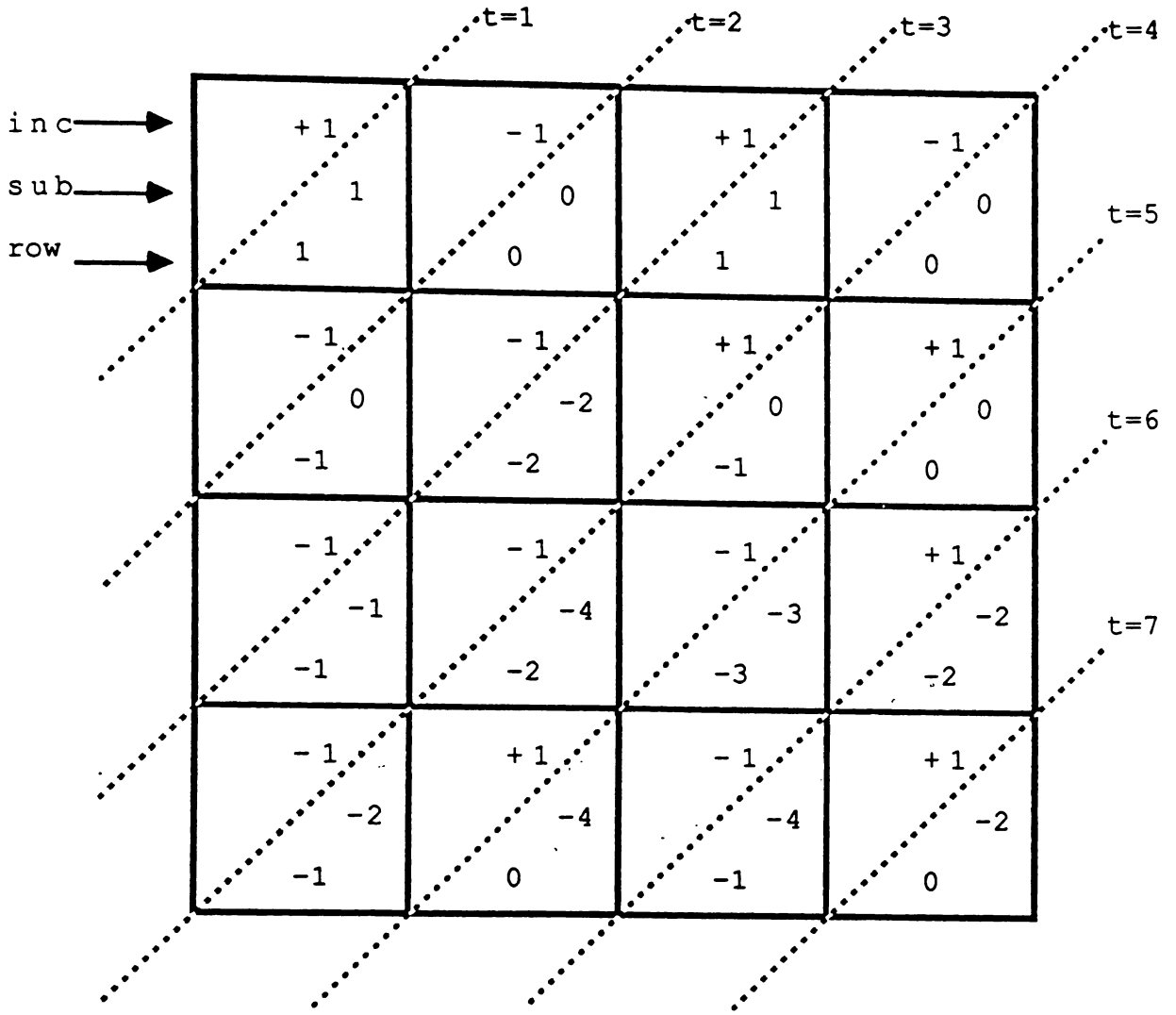

Figure 2. Computation of sub and row values in successive time units.

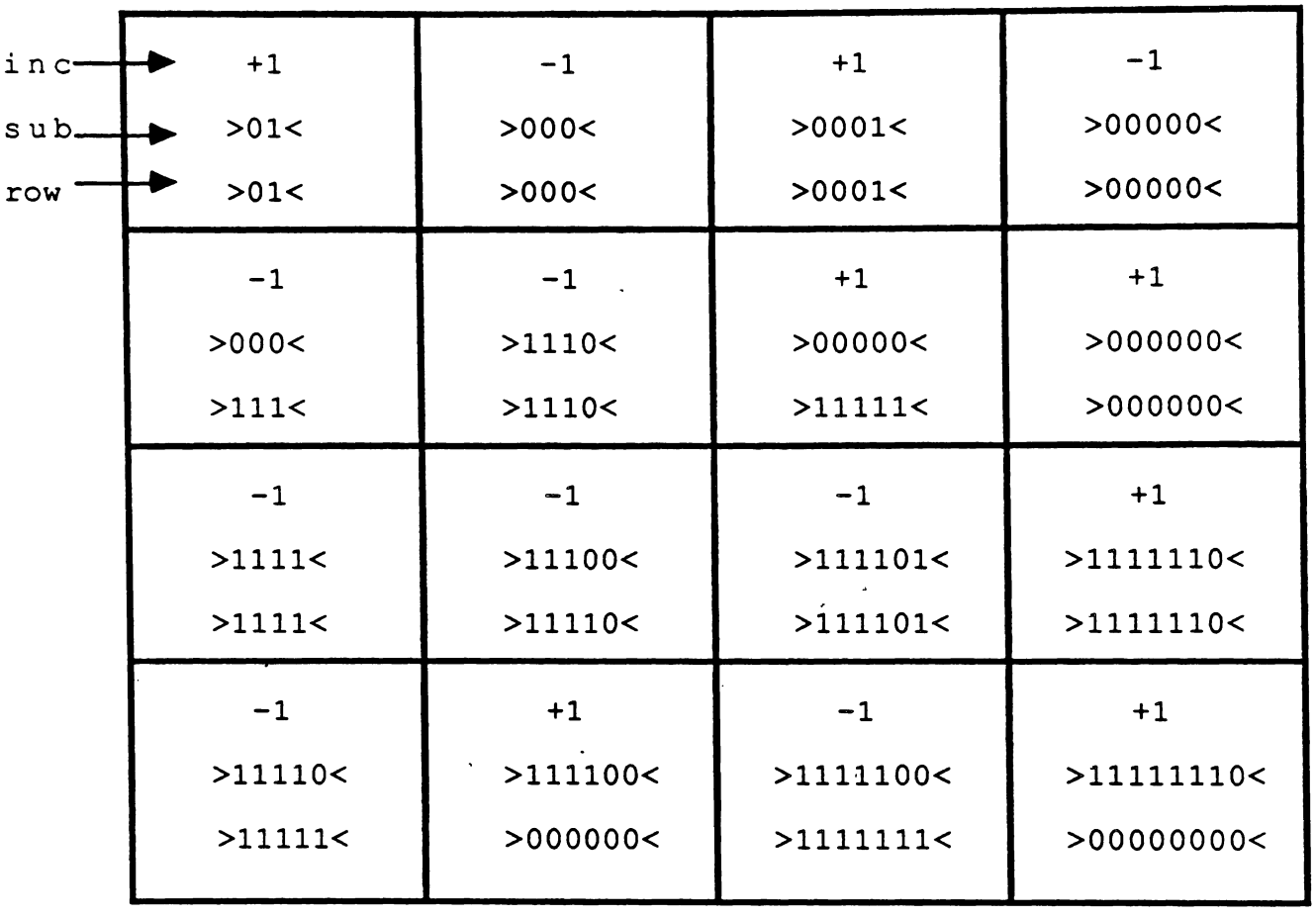

Figure 3. Two's-complement, (i+j)-bit representation of sub(i+j), row(i+j) values in Figure 2.

Time:	t-1	t	t+1	...	t+i+j-2	t+i+j-1	t+i+j	t+i+j+1
f_{i+j-1} (row value) in cell i,j-1	<	r_0	r_1	...	r_{i+j-2}	>		
f_{i+j-1} (sub value) in cell i-1,j	<	s_0	s_1	...	s_{i+j-2}	>		
f_{i+j} (sub value)	-	<	x_0	...	x_{i+j-3}	x_{i+j-2}	x_{i+j-1}	>
f_{i+j} (row value) in cell i,j	-	<	y_0	...	y_{i+j-3}	y_{i+j-2}	y_{i+j-1}	>

Figure 4. Bitwise computation of sub and row values in cell (i,j).

Time	Initial state zero			Initial state one		
	sub	row	answer	sub	row	answer
1	<	<	reject	<	<	reject
2	1	0	reject	1	0	reject
3	0	0	reject	1	0	reject
4	>	>	reject	>	>	accept

Figure 5. Starting the computation in cell (1,1)