JOURNAL OF COMPUTATIONAL PHYSICS 101, 176–184 (1992)

General Algorithm for Two-Dimensional Totalistic Cellular Automata

FRANCO BAGNOLI

Dipartimento di Matematica Applicata, Università di Firenze, Firenze, Italy, and INFN and INFM, Sezione di Firenze, Firenze, Italy

RAÚL RECHTMAN

Departamento de Física, Facultad de Ciencias, UNAM, Apdo. Postal 70-542, 04510 México D.F., México

AND

STEFANO RUFFO

Dipartimento di Energetica, Università di Firenze, Firenze, Italy, and INFN and INFM, Sezione di Firenze, Firenze, Italy

Received February 16, 1990; revised March 28, 1991

Multi-site coding techniques allow fast simulations of cellular automata that are economical in the use of memory. In these techniques the transition rule must be expressed using only bitwise operations. We present an algorithm for the simulation of generic totalistic and outer totalistic cellular automata which uses a multi-site coding technique. The algorithm is based on the careful use of (a) improvements over the canonical forms by using the exclusive-or operation, (b) optimal storage of the configuration in the computer memory, and (c) appropriate construction of stochastic rules. Items (b) and (c) of the method can be also applied to non-totalistic automata in any dimension.

1. INTRODUCTION

The study of cellular automata behavior, both of deterministic and probabilistic ones, is a subject of great interest nowadays [1]. Cellular automata are dynamical systems in which time, space, and dynamical variables are discrete. The space is a (regular) lattice, and each site (cell) takes a value in a discrete set. In this paper we restrict this set to $\{0, 1\}$, i.e., we consider only Boolean cellular automata. All the cells in the system evolve synchronously according to an uniform short-ranged law. The law gives the future state of a cell according to the present state of the cells belonging to a certain neighborhood. In a square lattice a widely used neighborhood is formed by the cell itself, the four nearest-neighbor ones, and the four next-to-nearest ones. This is often called the Moore neighborhood. It can also be divided into an outer neighborhood, formed by the eight cells surrounding the central one, and the cell itself.

Among the various automata, totalistic ones seem to represent a subset of limited extension (in two dimensions and with a Moore neighborhood there are 512 different totalistic rules with respect to 2^{512} general rules) that retains the complexity of the whole set [2]. The transition rule for totalistic cellular automata depends only on the sum of the cell values in the neighborhood. The class of totalistic rules is equivalent to that of the rules symmetric in all the arguments [3]. The transition rule for outer totalistic cellular automata depends separately on the value of the cell itself and on the sum of those in the neighborhood. Examples of such rules are Conway's Game of Life [4], biased majority rules that simulate interface motions [5], solidifications and aggregations models [6], and Ising dynamics [7, 8, 10].

In studying the statistical properties of these automata, long simulations of large arrays are often needed, requiring both powerful computers and big memory storage. The Multi-Site Coding technique (MSC) allows a gain in memory requirements and in execution speed [9, 10]. The main idea of MSC is to pack several variables into a single memory word of the computer (a word can hold from 16 to 64 bits, depending on the machine) and to elaborate their future value as a whole. In such a way a certain degree of parallelism can be achieved even on a serial computer. The drawback of this technique is how to implement a generic transition rule.

Storing a cell value into a single word allows its easy manipulation and the description of the rule with high-level language, such as: *if the sum of the neighbors is three, then....*

0021-9991/92 $5.00

Alternatively the cells values can be combined and used as an address in a precompiled look-up table. On the other hand MSC is fully exploited if the transition rule is expressed by means of operations acting over the packed variables as a whole. This can be achieved using the bitwise operations *NOT*, *AND*, *OR*, and eXclusive *OR* (*XOR*). The starting point is the canonical disjunctive form built directly from the truth table [11]. The canonical form for totalistic cellular automata contains a great number of operations. The reduction of this form to a minimal one (in the sense of the number of operations required) is not an easy task. In fact the problem of finding the minimal form is believed to be a NP one [12].

In this paper we discuss several aspects related to the simulation of cellular automata using the MSC technique. In Section 2 we present an algorithm that allows us to construct a generic two-dimensional totalistic and outer totalistic Boolean cellular automaton rule. The Boolean expressions obtained in this way improve significantly the canonical form. In Section 3 we discuss the problem of optimal storage of the configuration in the computer memory. The technique there exposed can be also applied to non-totalistic and non-Boolean cellular automata in any dimension. Section 4 briefly discusses the extension of the method to probabilistic rules; once more the results can be applied to the simulation of a generic stochastic cellular automaton. The subsequent section presents benchmarks among various implementations of the code and one application to the Game of Life. Section 6 presents some applications of these methods to physics, and the final section contains some conclusions.

2. THE ALGORITHM

In the following $x_{j,i}$ indicates the spin (cell value) at the site located at row j and column i in a square 2D lattice. Spins can take the values 0 and 1, so each site variable can be stored in a bit. Memory words are indicated with uppercase letters, as W, and arrays of words by $W_{j,k}$. All the array indices start from 0. The bits in a word are indicated with lowercase letters, thus

$$W = |w_{Nb-1}, ..., w_1, w_0|,$$

where Nb is the number of bits in a word (16, 32, or 64). Note that the order of the bits in a word is that required to read them as a number in base 2 in the standard left to right way.

The number of required words to store a row of the configuration is denoted by Nw and the number of sites in a row by Ns. Then

$$Ns = Nb \cdot Nw;$$

the number of rows required is indicated by Nr.

Given a site and its Moore neighborhood, the spin of the center is usually denoted by c and the spins of the neighbors by *nw*, *n*, *ne*, *w*, *e*, *sw*, *s*, *se*. The notation recalls north, west, east, and south directions. In the previous notation c is $x_{j,i}$, *nw* is $x_{j-1,i-1}$, etc. Following Vichniac's notation [7], we write

$$\begin{aligned} h &= nw + n + ne + w + e + sw + s + se \\ m &= h + c. \end{aligned} \tag{1}$$

Any totalistic evolution rule can be written as

$$c'(m) = \sum_{k=0}^{9} r_k \cdot m_k \tag{2}$$

and any outer totalistic rule as

$$c'(h, c) = \sum_{k=0}^{8} h_k \cdot [c \cdot r_{1,k} + (1-c) \cdot r_{0,k}]. \tag{3}$$

In these expressions c' is the updated value of the central site, m_k is 1 if $m = k$ and 0 otherwise and similarly for h_k, with m and h given by (1). Thus only one term contributes in the sums of Eqs. (2) and (3). The quantities r_k and $r_{c,k}$, $(c = 0, 1)$ take the value 0 or 1 and define the automaton rule.

As mentioned earlier it is advantageous to use MSC. In order to perform operations on all the bits in a word at once we need to use only bitwise operations. In the following we use the upper bar for the bit by bit negation, and the $\oplus$, $\wedge$, and $\vee$ symbols, respectively, for the bitwise *XOR*, *AND*, and *OR* operations. Let C denote a word that contains Nb spin variables. For the moment it is not important what the correspondence among the spins in C and the sites in the lattice is, as long as it is one-to-one. With the neighbors *nw*, *n*, *ne*, *w*, *e*, *sw*, *s*, *se* of each site stored in C, the neighbors words *NW*, *N*, *NE*, *W*, *E*, *SW*, *S*, *SE* can be constructed. Then, any totalistic rule may be written using MSC as

$$C' = \bigvee_{k=0}^{9} R_k \wedge M_k, \tag{4}$$

and any outer totalistic rule as

$$C' = \bigvee_{k=0}^{8} H_k \wedge [C \wedge R_{1,k} \vee \bar{C} \wedge R_{0,k}], \tag{5}$$

where R_k and $R_{c,k}$ are words whose bits are all equal to r_k and $r_{c,k}$, respectively. The words M_k and H_k contain in the ith bit $(i = 0, ..., Nb-1)$ the values of m_k and h_k for the neighborhood of the ith site packed in the word C.

The problem is now reduced to that of contructing the quantities m_k and h_k using only Boolean operations out of the spin c of the central site and of the spins *nw*, ..., *se* of the neighbors. Then, as mentioned above, the same Boolean expression may be applied in a bitwise fashion to *C*, *NW*, ..., *SE*. In what follows these expressions are constructed explicitly for totalistic rules and they are reported in table I; the case of outer totalistic rules is similar, and the corresponding expressions are summarized in Table II.

To simplify the notations let $y_0, y_1, ..., y_8$ denote the spins *c*, *nw*, ..., *se*. If in a certain configuration $(y_0, y_1, ..., y_8)$ the sum of the spins is m, the sum of the negations of the spins will be $9 - m$. This means that if expressions are found for m_k, $k = 5, ..., 9$, then the same formulas applied to the negations of all the spins will yield expressions for the other m_k with $k = 4, ..., 0$. As an example, m_9 takes the value 1 only if all the spins are 1, while m_0 is 1 only if all the spins are 0. Then

$$m_9 = \bigwedge_{i=0}^{8} y_i \tag{6}$$

$$m_0 = \bigwedge_{i=0}^{8} \overline{y_i}. \tag{7}$$

For $k = 8$ we have

$$m_8 = \bigvee_{\text{cyclic}} \overline{y_0} \wedge y_1 \wedge y_2 \wedge y_3 \wedge y_4 \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8, \tag{8}$$

TABLE I

Equivalent Configurations and Characteristic Functions for Totalistic Neighborhood

m	Independent conf.	Code	Characteristic function
9	111111111	511	$y_0 \wedge y_1 \wedge y_2 \wedge y_3 \wedge y_4 \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8$*
8	011111111	255	$(y_0 \oplus y_1) \wedge y_2 \wedge y_3 \wedge y_4 \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8$
7	001111111	127	
	010111111	191	
	011011111	223	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_3) \wedge y_4 \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8$
	011101111	239	$(y_0 \oplus y_1) \wedge y_2 \wedge (y_3 \oplus y_4) \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8$
	Total (107/395): $(y_0 \oplus y_1) \wedge [(y_2 \oplus y_3) \wedge y_4 \vee y_2 \wedge (y_3 \oplus y_4)] \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8$		
6	001011111	95	
	001101111	111	
	001111101	125	
	001111011	123	
	010101111	175	
	010110111	183	
	010111011	187	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_3) \wedge (y_4 \oplus y_5) \wedge y_6 \wedge y_7 \wedge y_8$
	001110111	119	
	000111111	63	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_4) \wedge (y_3 \oplus y_5) \wedge y_6 \wedge y_7 \wedge y_8$
	011011011	219	$(y_0 \oplus y_1) \wedge y_2 \wedge (y_3 \oplus y_4) \wedge (y_5 \oplus y_6) \wedge y_7 \wedge y_8$
	Total (161/1007): $(y_0 \oplus y_1) \wedge \{[(y_2 \oplus y_3) \wedge (y_4 \oplus y_5) \vee (y_2 \oplus y_4) \wedge (y_3 \oplus y_5)] \wedge y_6 \vee y_2 \wedge (y_3 \oplus y_4) \wedge (y_5 \oplus y_6)\} \wedge y_7 \wedge y_8$		
5	001010111	87	
	001011011	91	
	001011101	93	
	001100111	103	
	001101011	107	
	001101101	109	
	001110101	117	
	010101011	171	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_3) \wedge (y_4 \oplus y_5) \wedge (y_6 \oplus y_7) \wedge y_8$
	001001111	79	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_3) \wedge (y_4 \oplus y_6) \wedge (y_5 \oplus y_7) \wedge y_8$
	000011111	31	
	000101111	47	
	000110111	55	
	000111011	59	
	000111101	61	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_5) \wedge (y_3 \oplus y_6) \wedge (y_4 \oplus y_7) \wedge y_8$
	Total (170/1637): $(y_0 \oplus y_1) \wedge \{(y_2 \oplus y_3) \wedge [(y_4 \oplus y_5) \wedge (y_6 \oplus y_7) \vee (y_4 \oplus y_6) \wedge (y_5 \oplus y_7)] \vee (y_2 \oplus y_5) \wedge (y_3 \oplus y_6) \wedge (y_4 \oplus y_7)\} \wedge y_8$		

Note. The (*) at the end of the characteristic function for $m = 9$ means that the sum (*OR*) over all the cyclic translations is unnecessary. Total expressions are not reported for $m = 9$ and $m = 8$, where only one characteristic function is present. The numbers in brackets beside the total expressions are the required operations with respect to the canonical form.

where the *OR* operation is taken over the possible nine cyclic translations of the indices of the spins $y_0, y_1, \ldots, y_8$. Each term in the sum contains only one negated spin. Expression (8) may be rewritten as

$$m_8 = \bigvee_{\text{cyclic}} (y_0 \oplus y_1) \wedge y_2 \wedge y_3 \wedge y_4 \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8, \tag{9}$$

where $\oplus$ denotes the exclusive *OR*(*XOR*) operation. There is some redundancy in the last espression, as $a \oplus b = \bar{a} \wedge b \vee a \wedge \bar{b}$, so that each term in the sum already contains part of the subsequent term, but this saves one computer operation per term.

The configurations with seven spin variables equal to 1 fall into four classes. The elements of each class are equivalent under cyclic translations. Each class may be identified by the configuration that has the minimum code when read as a binary number. The equivalence classes and their code are shown in Table I. In this case the expression

$$(y_0 \oplus y_1) \wedge (y_2 \oplus y_3) \wedge y_4 \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8$$

is equal to 1 for the classes denoted by 127, 191, and 223 (eventually after a cyclic translation). The class 239 can be represented by

$$(y_0 \oplus y_1) \wedge y_2 \wedge (y_3 \oplus y_4) \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8.$$

Then

$$m_7 = \bigvee_{\text{cyclic}} (y_0 \oplus y_1) \wedge [(y_2 \oplus y_3) \wedge y_4 \vee y_2 \wedge (y_3 \oplus y_4)] \wedge y_5 \wedge y_6 \wedge y_7 \wedge y_8. \tag{10}$$

This expression contains a total of 107 Boolean operations, compared to the canonical form that contains 395. These numbers are also reported in Table I.

The expressions for m_6 and m_5 are obtained in a similar way and are shown in Table I. The efficiency of the algorithm increases with the number of configurations involved. The expressions for m_6 contain 161 operations in comparison to the canonical form that contains 1007. For m_5 the numbers are 170 and 1637, respectively. In order to

TABLE II

Equivalent Configurations and Characteristic Functions for Outer Totalistic Neighborhood

h	Independent conf.	Code	Characteristic function
8	11111111	255	$y_0 \wedge y_1 \wedge y_2 \wedge y_3 \wedge y_4 \wedge y_5 \wedge y_6 \wedge y_7$*
7	01111111	127	$(y_0 \oplus y_1) \wedge y_2 \wedge y_3 \wedge y_4 \wedge y_5 \wedge y_6 \wedge y_7$
6	00111111	63	
	01011111	95	
	01101111	111	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_3) \wedge y_4 \wedge y_5 \wedge y_6 \wedge y_7$
	01110111	119	$(y_0 \oplus y_1) \wedge y_2 \wedge (y_3 \oplus y_4) \wedge y_5 \wedge y_{67} y_7$
	Total (87/279): $(y_0 \oplus y_1) \wedge [(y_2 \oplus y_3) \wedge y_4 \vee y_2 \wedge (y_3 \oplus y_4)] \wedge y_5 \wedge y_6 \wedge y_7$		
5	00101111	47	
	00110111	55	
	00111011	59	
	00111101	61	
	01010111	87	
	01011011	91	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_3) \wedge (y_4 \oplus y_5) \wedge y_6 \wedge y_7$
	00011111	31	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_4) \wedge (y_3 \oplus y_5) \wedge y_6 \wedge y_7$
	Total (96/615): $(y_0 \oplus y_1) \wedge [(y_2 \oplus y_3) \wedge (y_4 \oplus y_5) \vee (y_2 \oplus y_4) \wedge (y_3 \oplus y_5)] \wedge y_6 \wedge y_7$		
4	00101011	43	
	00101101	45	
	00110011	51	
	00110101	53	
	01010101	85	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_3) \wedge (y_4 \oplus y_5) \wedge (y_6 \oplus y_7)$
	00100111	39	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_3) \wedge (y_4 \oplus y_6) \wedge (y_5 \oplus y_7)$
	00001111	15	
	00010111	23	
	00011011	27	
	00011101	29	$(y_0 \oplus y_1) \wedge (y_2 \oplus y_5) \wedge (y_3 \oplus y_6) \wedge (y_4 \oplus y_7)$
	Total (135/839): $(y_0 \oplus y_1) \wedge \{(y_2 \oplus y_3) \wedge [(y_4 \oplus y_5) \wedge (y_6 \oplus y_7) \vee (y_4 \oplus y_6) \wedge (y_5 \oplus y_7)] \vee (y_2 \oplus y_5) \wedge (y_3 \oplus y_6) \wedge (y_4 \oplus y_7)\}$		

Note. The (*) at the end of the characteristic function for $h = 8$ means that the sum (*OR*) over all the cyclic translations is unnecessary. As in Table I total expressions for $h = 8$ and $h = 7$ are not reported. The numbers in brackets beside the total expressions are the required operations with respect to the canonical form.

achieve the maximum execution speed, the sum over the cyclic translations in the expressions reported in Table I should be explicitly developed.

In a general algorithm all the quantities M_k have to be generated, as the selection of the rule is performed by the R_k. It is possible to save the task of generating m_4 and m_5 using the parity p of the sum m, given by

$$p = y_0 \oplus y_1 \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_7 \oplus y_8; \quad (11)$$

p is 0(1) if m is even (odd). Obviously

$$p = m_1 \vee m_3 \vee m_5 \vee m_7 \vee m_9,$$

and since

$$m_i \wedge m_j = 0 \qquad \text{if} \quad i \neq j,$$

we get

$$m_5 = p \wedge \overline{m_1 \vee m_3 \vee m_7 \vee m_9}. \quad (12)$$

The expression for m_4 may be obtained from the condition

$$\bigvee_{k=0}^{9} m_k = p \vee \bar{p} = 1;$$

then

$$m_4 = \bar{p} \wedge \overline{m_0 \vee m_2 \vee m_6 \vee m_8}. \quad (13)$$

The expressions (11), (12), and (13) imply only 19 operations, with respect to the 3274 of the canonical form and of the 340 of the reduced form of Table I. In the actual writing of the algorithm in a computer code one can further reduce the number of operations by taking into account the presence of common patterns in the expressions in Tables I and II and observing that $a \oplus b = \bar{a} \oplus \bar{b}$. In total the number of operations required to implement a generic totalistic automaton is about 600 bitwise operations per word.

Reasoning in a similar fashion, one may find compact expressions for outer totalistic cellular automata. These are presented in Table II.

3. IMPLEMENTING THE ALGORITHM

The full power of the algorithm is developed when applied to full words. There are several ways in which one may assign the spins of the sites $x_{j,i} (j = 0, ..., Nr - 1; i = 0, ..., Ns - 1)$ in the lattice to the words $X_{j,k} (k = 0, ..., Nw - 1)$; however, the task of building the neighborhood of the sites stored in the word $C = X_{j,k}$ must be as economical as possible. The final goal is to have the values of the cells belonging to the neighborhood of a cell stored in a certain position of the word C in the corresponding bits of the words $NW, ..., SE$. This can be obtained without any shift operations by assigning the first spin in a row to the first bit of the first word, the second spin to the first bit of the second word, and so on for the first NW spins. The previous operation is repeated Nb times in order to store the first Nw spins in the first bits of the words containing the row, the following Nw spins in the second bits of the words, and so on. For a generic row j we have

$$\begin{aligned} X_{j,0} &= |x_{j,(Nb-1)\cdot Nw}, ..., x_{j,2\cdot Nw}, x_{j,Nw}, x_{j,0}| \\ X_{j,1} &= |x_{j,(Nb-1)\cdot Nw+1}, ..., x_{j,2\cdot Nw+1}, x_{j,Nw+1}, x_{j,1}| \\ &\cdots \\ X_{j,Nw-1} &= |x_{j,Nb\cdot Nw-1}, ..., x_{j,3\cdot Nw-1}, x_{j,2\cdot Nw-1}, x_{j,Nw}|. \end{aligned} \quad (14)$$

For $Nw \geq 3$ and apart from boundary conditions, the spins of the neighbors of the sites in $X_{j,k}$ are stored in the corresponding bits of the words

$$\begin{pmatrix} NW_{j,k} & N_{j,k} & NE_{j,k} \\ W_{j,k} & C_{j,k} & E_{j,k} \\ SW_{j,k} & S_{j,k} & SE_{j,k} \end{pmatrix} = \begin{pmatrix} X_{j-1,k-1} & X_{j-1,k} & X_{j-1,k+1} \\ X_{j,k-1} & X_{j,k} & X_{j,k+1} \\ X_{j+1,k-1} & X_{j+1,k} & X_{j+1,k+1} \end{pmatrix}. \quad (15)$$

In order to implement periodic boundary conditions on the horizontal border, all the operations on the index j are to be considered modulus the number of the rows Nr. Vertical periodic boundary conditions are imposed by observing that the west neighbors of the sites in the first word $X_{j,0}$ are contained in the last word $X_{j,Nw-1}$ circularly shifted one bit to the left, and the east neighbors of the spins in $X_{j,Nw-1}$ are in $X_{j,0}$ circularly shifted one bit to the right. This storage scheme can be used in any dimension and even for non-Boolean automata such as lattice gases (see Section 6); with a few modifications it can also be adapted to larger neighborhoods.

4. PROBABILISTIC TOTALISTIC CELLULAR AUTOMATA

Probabilistic cellular automata may be implemented by allowing real values between 0 and 1 for the coefficients r_k of Eq. (2) and interpreting c' as the probability that the spin of the central site assumes the value 1 at the next time step. Then r_k is the probability that this spin is 1 if m is equal to k.

These probabilistic concepts may be introduced in the bitwise evolution rule (4) filling the bit masks R_k with bits having the value 1 with probability r_k. A large number Nm of samples of the words $R_{k,x}$ is constructed with

$x = 0, ..., Nm - 1$. Then, given a random number x between 0 and $Nm - 1$, the evolution rule for the sites in a word C is given by

$$C' = \bigvee_{k=0}^{9} R_{k,x} \wedge M_k. \tag{16}$$

In practice it is possible to reduce the number Nm of the independent random masks by performing a random circular shift over $R_{k,x}$ before introducing it in Eq. (16). The same arguments apply to outer totalistic cellular automata. Once again, the use of random masks can easily be adapted to other evolution rules. An advantage in using predefined random masks is that the probability can be fixed with a great precision, and correlations are further depressed by shuffling the random masks.

5. PERFORMANCES

In order to obtain an accurate estimate of the time required per site update, we propose an approximate expression for the running time T of a program,

$$T = t_l + t_i \cdot L + t_u \cdot L \cdot N_u, \tag{17}$$

where L is the number of lattice sites, N_u is the number of global updates of the lattice, and t_l, t_i, t_u are constants: t_l represents the loading time, which could also involve the compilation time; t_i is the time requested to initialize the lattice and t_u is the update time per site. The time needed to implement the periodic boundary conditions is not considered, but the linearity of the law with respect to L with fixed N_u has been tested for lattice sizes ranging from 64×64 to 512×512 sites, over a variety of machines. The quantity used to compare the performances of the various implementations is the number ν of sites updated in a second, and it is obtained by

$$\nu = \left\langle \frac{L \cdot \Delta N_u}{\Delta t} \right\rangle, \tag{18}$$

where ΔN_u and Δt represent the differences in lattice updates and in running time of two samples of the same program on the same machine for different N_u's. The angular brackets represent the average over different samples.

The computers used for the benchmarks were an IBM PS/2 80 with a clock speed of 16 MHz (using DOS operating system the words are 16 bits long); a SUN 386*i* workstation at 25 MHz, a VAX 3580 (32 bits per word), an HP 9000/840 (32 bits per word), and a CRAY I XMP (64 bits per word). All the programs were written in a high-level language and in a clear style with many calls to subroutines and no dirty tricks. Vectorialization on the CRAY was explicitly avoided, since we were interested in testing the gain obtained with the reduced rule on different machines. The central loop containing the sum (OR) over the eight cyclic translations of the indices was not explicitly developed, and the variables $x_0, x_1, ..., x_8$ were translated by explicit assignment (temp $= x_0$, $x_0 = x_1$, ..., $x_8 =$ temp).

In order to show the advantage represented by using the algorithm described above with respect to the canonical form, eight different programs were written using the C language and run on the HP computer (Table III). The table shows the differences in the number of spins updated in a second, between the canonical expressions and our algorithm in constructing all the quantities m_k and h_k. The performances are reported for inline code and for a structured call to a subroutine. Our algorithm is about three times faster than the canonical form, depending on the complexity of the calculation.

The algorithm may be applied to many interesting models. The efficiency is discussed briefly in two cases: Conway's Game of Life [2], which is a classical testing ground, and a general probabilistic outer totalistic rule. For outer totalistic rules, and in particular for the Game of Life, we should expect smaller improvements with respect to the canonical form due to the simplicity of the rule (see Table II). When not explicitly indicated, the following programs were written in FORTRAN 77.

The Game of Life is a two-dimensional outer totalistic cellular automaton whose evolution rule is given by

$$c' = \begin{cases} 1 & \text{if } c = 0 \text{ and } h = 3, \\ 1 & \text{if } c = 1 \text{ and } h = 2 \text{ or } 3, \\ 0 & \text{otherwise,} \end{cases} \tag{18}$$

where h is the sum of the neighbors of the central cell $x_{i,j}$ as defined in (1). It should be noted from expression (18) that only m_2 and m_3 are required in order to calculate c'.

We wrote three different programs:

1. *High-level*, a traditional code with one spin per word and the rule implemented with *if... then...* statements;

TABLE III

Number ν of sites updated in a second in kHz (formula (18)) for the construction of m_k (totalistic) and h_k (outer totalistic) using C language on a HP 9000/840 computer

	Totalistic		Outer totalistic	
Program	Subroutine	Inline	Subroutine	Inline
Canonical	83.0	172.0	177.8	338.6
Reduced	301.6	441.8	447.1	659.5

TABLE IV

Number v of sites updated in a second in kHz (formula (18)) for the Game of Life

Computer	High-level	Canonical	Reduced
PS 2/80	18.9	3.5	22.5
SUN 386*i*	32.8	50.4	174.5
VAX 8350	54.9	117.8	217.2
HP 9000/840	101.9	175.1	585.1
CRAY I XMP	391.3	3034.1	4551.1

2. *Canonical*, using MSC and the rule implemented via the canonical disjunctive form;

3. *Reduced*, using MSC and the compact expressions reported in Table II.

The values of v obtained running the three different programs are reported in Table IV. The reduced program always runs faster than the other two. We observe that when the number of bits per word is small, as in the case of the PS/2, the use of MSC does not necessarily imply better performances. For computers with more bits per word the gain of MSC increases, but still one gets significant improvements using our algorithm.

Finally, we implemented a code for the simulation of a general outer totalistic probabilistic rule. The results for v are reported in Table V. The gain with respect to the high-level program is less than above (apart the results for the HP computer), even if with MSC only one random number has to be extracted for every Nb spins. This is due to the larger number of operations needed to generate all the M_k in (5) with respect to the Game of Life.

In order to evaluate the influence of the language used, a program that implements a general outer totalistic probabilistic rule was written in C language for the HP computer. The update rate obtained was 257.2 kHz, which is nearly 1.5 times the speed of the corresponding FORTRAN program. This result cannot be generalized, but it shows that several factors should contribute in order to achieve the best performances.

A preliminary study of probabilistic mixtures of totalistic automata has been undertaken. An example is the mixture

TABLE V

Number v of sites updated in a second in kHz (formula (18)) for a general probabilistic outer totalistic rule

Computer	High-level	Reduced
PS 2/80	9.1	9.5
SUN 386*i*	16.6	73.1
VAX 8350	41.4	93.1
HP 9000/840	15.2	176.17
CRAY I XMP	261.2	2239.3

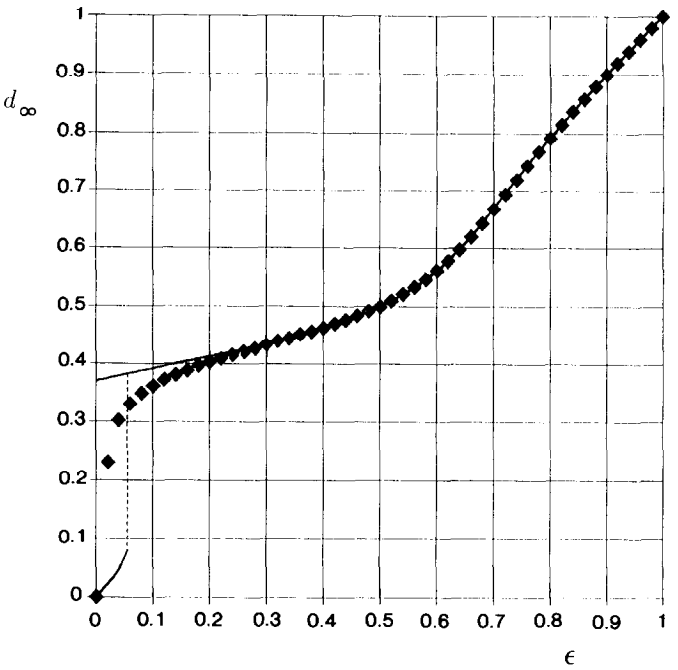

FIG. 1. Asymptotic density d_∞ of live cells for a random mixture of the Game of Life and its conjugation with probability ε varying from 0 to 1 step 0.02. We plot the result of computer simulations (diamonds) and of mean field approximation [14] (continous line). At $\varepsilon = 0$, which corresponds to the Game of Life, the asymptotic density is $d_\infty = 0.028$.

of the Game of Life with its conjugate that assumes the value 0(1), whereas the Game of Life takes the value 1(0). At any site the Life rule is applied with probability ε and its conjugate with probability $1-\varepsilon$ ($0 \leqslant \varepsilon \leqslant 1$), as described in Section 4. In Fig. 1 the asymptotic density of live sites (ratio of sites having spin equal to 1 to the total number of sites) is shown as a function of ε, together with a mean field approximation [14]. These simulations were carried out on a 256×256 lattice. The graph shows 50 points, each being the average over 10 simulations performed on the SUN 386*i* with a program written in C. The total CPU time was roughly 36 h. The update speed was 161.6 kHz, which is smaller than the value reported in Table IV due to the loading time, the initialization time, and the (relatively small) slowing down due to the calculation of the density every 100 updates performed to monitor the relaxation of the lattice. The algorithm here described was also used to study the relaxation and the critical properties of the Game of Life in Ref. [15].

6. SOME PHYSICAL APPLICATIONS

As a first physical example let us consider the simple model of interface motion developed in Ref. [5], based on a marginal majority deterministic rule. The central site c assumes the value that is most prevalent in its Moore neighborhood only if the majority is strong ($m \geqslant 6$, see

Eq. (1)), or in case of a marginal minority ($m=4$). In terms of Eq. (2) the rule is defined as

$$\begin{aligned} r_k &= 1 \qquad \text{if} \quad k = 4, 6, 7, 8, 9 \\ r_k &= 0 \qquad \text{otherwise.} \end{aligned} \tag{19}$$

The twist in the majority provides a kind of frustration that simulates a mobile interface according to the Allen–Cahn equation.

Rule (19) can be coded with the general algorithm described in Section 2, building the R_k of Eq. (4) and then using the expression for the m_k of Table I and of Eqs. (11), (12), and (13). Of course, for this given rule, one can further reduce the number of required operations using ad hoc tricks; in the case of rule (19) we were able to derive an expression containing 42 operations.

The full power of the algorithm is developed when applied to probabilistic cellular automata.

Let us first discuss as a simple example the application of the method to a parallel Monte-Carlo simulation of an Ising model with nearest and next-to-nearest neighbors equal interactions and zero magnetic field. The Hamiltonian $\mathscr{H}$ is

$$\mathscr{H} = \sum_{j,i} -J\sigma_{j,i}\left[\left(\sum_{\substack{l=j-1,j,j+1 \\ k=i-1,i,i+1}} \sigma_{l,k}\right) - \sigma_{j,i}\right],$$

where $\sigma_{j,i} = \pm 1$ and $J \geqslant 0$ is a ferromagnetic coupling. Passing to Boolean variables ($\sigma_{j,i} = 2x_{j,i} - 1$), the local energy H is given by

$$H(c, h) = -2J(2c-1)(h-4),$$

where $c = x_{j,i}$ is the state of a generic site and h is defined as in Eq. (1).

The variation of the local energy with the flip of c is

$$\Delta H(c, h) = H(\bar{c}, h) - H(c, h) = \mp 4J(h-4),$$

where the minus (plus) sign holds for the transition of c from 0 (1) to 1(0).

We can now define the coefficient $r_{c,h}$ of Eq. (3) so that the transition probabilities satisfy detailed balance,

$$r_{c,h} = \begin{cases} 1 & \text{if} \quad \Delta H \leqslant 0, \\ \exp -\dfrac{\Delta H}{kT} & \text{if} \quad \Delta H > 0. \end{cases}$$

The random masks are built as described in Section 4. The number of required random masks can be reduced in this case observing that ΔH is always an integer multiple of $4J$, so that the $r_{c,h}$ are integer powers of $p = \exp -4J/kT$. A word A whose bits are one with probability p^2 can be obtained from two independent random words B and C whose bits are one with probability p just performing a bitwise AND among them: $A = B \wedge C$.

The implementation of a parallel version of the Metropolis Monte-Carlo sampling technique must be very cautious, as the full parallelism can lead the system towards a maximum of the energy, as discussed in Ref. [7]. A detailed discussion of the simulations of the Ising model with cellular automata can be found in Ref. [16]. The same scheme can be used to simulate Ising models in spaces of higher dimensions, for instance in three dimensions with nearest neighbors interactions (six neighbors).

The storage scheme described in Section 3 and the introduction of probabilistic evolution rules of Section 4 can be also applied to non-totalistic cellular automata such as lattice gas models [17]. For example, to model diffusion as suggested in [18], one needs a 4-bit per site cellular automaton. To this aim let us consider four Boolean cellular automata lattices stacked one over the other. We refer to the four lattices (planes) as UP(subscript $\uparrow$), $LEFT(\leftarrow)$, $DOWN(\downarrow)$, and $RIGHT(\rightarrow)$. If, for instance, at site (j, i) the bit $x_\uparrow$ is set to one, it means that there is a particle travelling from site (j, i) to $(j-1, i)$. Diffusion is controlled by two parameters p and q, which give the probability of a common counter-clockwise rotation of all the particles in a generic site according to the scheme of Table VI. The random masks P_x and Q_x are built according to the parameters p and q.

The evolution rule for a word $C_\uparrow$ that contains Nb sites of the UP plane is

$$C'_\uparrow = \overline{Q_x} \wedge \overline{P_x} \wedge S_\uparrow \vee \overline{Q_x} \wedge P_x \wedge W_\rightarrow \vee Q_x \wedge \overline{P_x} \wedge E_\leftarrow \vee Q_x \wedge P_x \wedge N_\downarrow,$$

where N, W, S, E denote the words that contain the nearest neighbors of the spins in C. The expressions for the other

TABLE VI

Counter-clockwise rotations of the velocities of the particles in a site for a probabilistic lattice gas, according to the parameters p, q

p	q	Rotation
0	0	0
0	1	$-\frac{\pi}{2}$
1	0	$\frac{\pi}{2}$
1	1	π

planes are obtained by performing a cyclic permutation of the $\uparrow, \leftarrow, \downarrow, \rightarrow$ and S, E, N, W symbols.

7. CONCLUSIONS

A general algorithm for totalistic and outer totalistic cellular automata using MSC has been developed that allows significant reduction in execution times with respect to the canonical form. We have concentrated our efforts mainly in simplifying the rule, disregarding the problem of vectorization and of special implementations on parallel machines. A complementary approach dealing with these problems may be found in Ref. [13]. In general, the main drawback of MSC is the serialization needed to perform calculations; on the other hand, a great advantage is the efficient use of the memory. Our algorithm is applicable to any totalistic rule and may still be improved in specific applications. We have also discussed the problem of optimal storage of the configuration in the computer memory and the extension of the results to stochastic models. These last topics can be also applied to non-totalistic automata in any dimension. Finally, examples of physical applications are given. We have shown how to apply the algorithm to a deterministic totalistic rule that simulates interface motion between two fluids, to a Metropolis Monte-Carlo Ising model (a probabilistic outer totalistic rule), and to a stochastic lattice gas.

Note added in proof. One of the authors (F.B) has further developed the algorithm presented here in "Boolean derivatives and computation of cellular automata," *Int. J. Mod. Phys.* **C**, in press.

ACKNOWLEDGMENTS

The ideas we have presented have profited from useful discussions with R. Livi and A. Salcido. We also thank G. Vichniac for reading the manuscript and for suggesting to us the equivalence totalistic/symmetric of Ref. [3]. This work was partially done during reciprocal visits to Florence and Mexico City partially sponsored by the CNR of Italy and the CONACYT of Mexico.

REFERENCES

1. S. Wolfram (Ed.), *Theory and Applications of Cellular Automata*, (World Scientific, Singapore, 1986); and *Physica* **10D** (1984).
2. N. Packard and S. Wolfram, *J. Stat. Phys.* **38**, 901 (1985); also reprinted in [1].
3. This theorem is due to Shannon, see e.g., R. E. Miller, *Switching Theory*, Vol. I (Wiley, New York, 1966), p. 103.
4. M. Gardner, *Sci. Am.* **223**, 120 (1970); *Sci. Am.* **223**, 116 (1970); *Sci. Am.* **224**, 104 (1971); *Sci. Am.* **224**, 112 (1971); *Sci. Am.* **224**, 114 (1971); *Sci. Am.* **226**, 104 (1972); and in *Life, Wheels and Other Mathematical Amusements*, edited by M. Gardner (W. H. Freeman, New York, 1983).
5. G. Vichniac, in *Chaos and Complexity*, edited by R. Livi, S. Ruffo, S. Ciliberto, and M. Buiatti (World Scientific, Singapore, 1988).
6. N. Packard, in *Proceedings of the First International Symposium for Science on Form*, edited by Y. Katoh *et al.* (KTK Scientific Publisher, 1986); also reprinted in [1].
7. G. Vichniac, *Physica* **10D**, 96 (1984).
8. Y. Pomeau, *J. Phys. A* **17**, L415 (1984).
9. R. Zorn, H. J. Herrmann, and C. Rebbi, *Comput. Phys. Commun.* **23**, 337 (1987); C. Kalle and V. Winkelman, *J. Stat. Phys.* **28**, 639 (1982); S. Wansleben, J. G. Zabolitzky, and C. Kalle, *J. Stat. Phys.* **37**, 271 (1984).
10. H. J. Herrmann, *J. Stat. Phys.* **37**, 271 (1984); J. G. Zabolitzky and H. J. Herrmann, *J. Comp. Phys.* **76**, 426 (1988).
11. S. Yablonski, *Introduction aux Mathématiques Discretes* (MIR, Moscow, 1983).
12. F. Wegener, *The Complexity of Boolean Functions* (Wiley, New York, 1987).
13. J. Myczkowski and G. Vichniac, "Parallel Programming for Cellular Automata," AICA Workshop on Parallel Programming, CINECA, Bologna, Oct. 1989 (unpublished).
14. L. S. Schulman and P. E. Seiden, *J. Stat. Phys.* **19**, 293 (1978).
15. F. Bagnoli, R. Rechtman and S. Ruffo, *Physica A* **171**, 249 (1991).
16. O. Parodi and H. Ottavi, in *Cellular Automata and Modeling of Complex Physical Sistems, Proceedings of the Winter School, Les Houches, France*, 1989, edited by P. Manneville, N. Boccara, G. Y. Vichniac, and R. Bideaux (Springer-Verlag, Berlin 1989), p. 82.
17. Gary D. Doolen (Ed.), Lattice Gas Methodes for PDE's, *Physica* **47D**, 1991.
18. B. Chopard and M. Droz, in *Cellular Automata and Modeling of Complex Physical Systems, Proceedings of the Winter School, Les Houches, France*, 1989, edited by P. Manneville, N. Boccara, G. Y. Vichniac and R. Bideaux (Springer-Verlag, Berlin, 1989), p. 130.