Article:
Billings, S.A. and Yang, Y.X. (2003) Identification of the neighborhood and CA rules from spatio-temporal CA patterns. IEEE Transactions on Systems Man and Cybernetics Part B: Cybernetics, 33 (2). pp. 332-339. ISSN: 1083-4419

https://doi.org/10.1109/TSMCB.2003.810438

Correspondence

Identification of the Neighborhood and CA Rules From Spatio–Temporal CA Patterns

S. A. Billings and Yingxu Yang

***Abstract*—Extracting the rules from spatio–temporal patterns generated by the evolution of cellular automata (CA) usually produces a CA rule table without providing a clear understanding of the structure of the neighborhood or the CA rule. In this paper, a new identification method based on using a modified orthogonal least squares or CA-OLS algorithm to detect the neighborhood structure and the underlying polynomial form of the CA rules is proposed. The Quine–McCluskey method is then applied to extract minimum Boolean expressions from the polynomials. Spatio–temporal patterns produced by the evolution of one-dimensional (1-D), two-dimensional (2-D), and higher dimensional binary CAs are used to illustrate the new algorithm and simulation results show that the CA-OLS algorithm can quickly select both the correct neighborhood structure and the corresponding rule.**

***Index Terms*—Cellular automata (CA), identification, spatio–temporal systems.**

I. Introduction

Cellular automata (CA) represents an important class of models that evolve in time over a spatial lattice structure of cells. CAs have been applied in image processing [1], pattern recognition [2], digital circuit design [3], and robotics [4]. Many authors have demonstrated that relatively simple binary CA rules can produce highly complex patterns of behavior. These results illustrate the potential of CAs as a model class and suggest that it may be possible to model even very complex spatio–temporal behavior using CA models of a simple form. However, very few studies have investigated how these rules can be extracted from observed patterns of spatio–temporal behavior.

Ideally, the identification technique should produce a concise expression of the rule. This ensures that the model is parsimonious and can readily be interpreted either for simulation or hardware realization of the CA. Sequential and parallel algorithms for computing the local transition table were presented by Adamatzky [5], and Richards [6] introduced a method using genetic algorithms (GAs). However, no clear structure of the related neighborhoods was obtained in either of these studies and the detection process was complicated. GAs were also employed in [7] to determine the rules as a set of logical operators. Parsimonious local rules were found for low-dimensional CAs, but when CA with large-size neighborhoods are involved the search process can be computationally demanding, sometimes taking several hours in a single run. This is mainly because of the nature of the GA evolution. If the CA model to be identified can be reconfigured into linear-in-the-parameters model, then the identification method would not have to be restricted to a GA related approach, and this may allow the development of a fast identification procedure. This is achieved in this paper.

Manuscript received February 13, 2000; revised July 11, 2000 and March 21, 2002. This work was supported by the University of Sheffield and the U.K. EPSRC. This paper was recommended by Associate Editor B. J. Oommen.

The authors are with the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield S1 3JD, U.K. (e-mail: s.billings@sheffield.ac.uk; yingxu@acse.shef.ac.uk).

Digital Object Identifier 10.1109/TSMCB.2003.810438

In this paper, a totally new approach is adopted to identify both the neighborhood and the CA rule from complex patterns of high-dimensional spatio–temporal behavior. Identifying the CA rule or model is considered as a two-stage procedure. First, the neighborhood which defines the spatial interaction of the cells over a temporal window is determined and then the rules that specify the values of these cells is estimated. Earlier studies [5]–[7] have attempted to devise solutions to these problems based on the logical rule base which defines binary CAs, but this involves determining the spatio–temporal rules as nonlinear combinations of cellular values.

In the present research, this problem is avoided by exploiting the fact that the binary rules can be expressed as Boolean functions and showing that these can be exactly represented using simple polynomial models. The main advantage of this is that now the problem is mapped into a linear-in-the-parameters model. A modified orthogonal least squares algorithm, called the CA-OLS method, is introduced which determines the neighborhood and the unknown model parameters. The Quine–McCluskey algorithm can then be applied to extract the minimum Boolean expression to produce the final CA model. Mapping the problem into a polynomial model form, determining the structure and parameters, and then mapping back to a logical expression produces for the first time a powerful method for determining the rules of high-dimensional CAs in the form of a parsimonious model. This is achieved from just the observations of the data and no *a priori* information.

II. Cellular Automata and the Difficulties of Cellular Automata

A cellular automaton is defined by three parts: 1) a neighborhood; 2) a local transition rule; and 3) a discrete lattice structure consisting of a large number of cells which are occupied by states from a finite set of discrete values. The local transition rule updates all cells synchronously by assigning to each cell, at a given time step, a value which depends only on the neighborhood. Attention in this paper is restricted to binary CA where the cells can only take binary values. Although binary CAs form one of the simplest classes of CAs, they have been the focus of most investigations and are capable of generating complicated patterns of global behavior and capturing the essential features of many complex phenomena.

A. CA Neighborhoods

The neighborhood of a cell is the set of cells over both space and time that are directly involved in the evolution of the cell. Sometimes this includes the cell itself. The neighborhood structure varies depending on the construction of the CA. Consider a one-dimensional (1-D) 3-site CA for example. Denoting the cell at position j at time step t as $\mathrm{cell}(j;t)$, then the neighborhood of $\mathrm{cell}(j;t)$ could be a von Neumann neighborhood, illustrated in Fig. 1(a), or the two exotic neighborhoods shown in Fig. 1(b) and (c), respectively. The neighborhood can involve cells from different spatial and temporal scales. The exotic neighborhood in Fig. 1(b) encompasses cells from the same temporal scale but different spatial scale than the cells in the von Neumann neighborhood while the neighborhood in Fig. 1(c) involves cells from the same spatial scale but different temporal scale from the cells in the von Neumann neighborhood. There are many more possible neighborhood structures for two-dimensional (2-D) CA. The most commonly used are the 5-site von Neumann

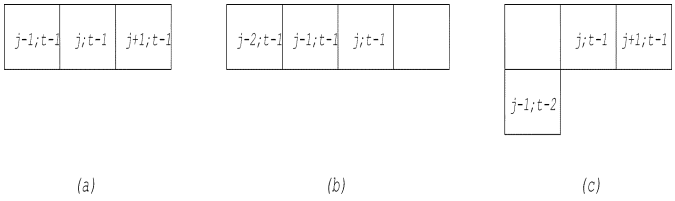

Fig. 1. Examples of 3-site neighborhoods for a 1-D CA. (a) von Neumann neighborhood. (b) and (c) Exotic neighborhoods.

neighborhood and the 9-site Moore neighborhood. The neighborhood structures for higher dimensional CA are much more complicated and diverse than the 1-D and 2-D cases. Neighborhoods for 2-D and higher dimensional CAs can also involve cells from temporal scales other than $t-1$. For example, a 5-site 2-D neighborhood could take the form $\{\mathrm{cell}(i,j-1;t-3),\mathrm{cell}(i,j-1;t-2),\mathrm{cell}(i,j-1;t-1),\mathrm{cell}(i+1,j;t-1),\mathrm{cell}(i,j+1;t-1)\}$. Clearly, the number of possible cells in a multidimensional CA over a range of temporal scales can be huge.

B. Local CA Transition Rules

Local transition rules can be defined in several equivalent ways. The most common method is to use a transition table analogous to a truth table where the first row describes the states of the neighborhood and the second row indicates the next state of the cells. The rules are then labeled by specifying which neighborhoods map to zero and which to 1. The standard form of a 3-site 1-D rule R is shown below

$$\begin{array}{cccccccc} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ r_0 & r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 \end{array}$$

where $r_i, i=0,\ldots,7$ indicates the next-states of the cells. Every component r_i corresponds to a coefficient 2^i which is essential in computing the numerical label associated with the rule. The numerical label D assigned to rule R above is therefore given by $D(R)=\sum_{i=0}^{2^3-1} r_i 2^i$, which is simply the sum of the coefficients associated with all the nonzero components. For example, the well-known 1-D 3-site rule *Rule22* is defined as $\mathrm{Rule22} = (01\,101\,000)$. The numerical label is given by $D(\mathrm{Rule22}) = 2^1+2^2+2^4 = 22$. The transition table of this rule is shown below

$$\begin{array}{cccccccc} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}.$$

C. Difficulties in CA Identification

Many studies of CA have focused on demonstrating that relatively simple CA rules can produce complex patterns of behavior. This demonstrates the potential of CAs as a model class but shows that this can only be realized if the simple underlying rules can be determined from observed spatio–temporal behavior.

When identifying CA rules the only *a priori* knowledge will be the spatio–temporal patterns produced by the evolution of the CA. Realistically, the neighborhood structure including the size will be unknown and this means that the possible combinations can number into the hundreds of millions. It may be possible to find simple CA rules by searching through the rule space when only 1-D or 2-D CAs with very small neighborhoods are involved. However, as the neighborhood size, or the dimensionality, or both increase the combinational possibilities become huge. The few authors that have studied this problem [5], [6] have therefore focused on a very limited class of low-dimensional CAs. However, there is a clear need to develop procedures which can operate on observed data from CAs over higher dimensional spatial and temporal neighborhoods with no *a priori* information. It is assumed that a sequence of the CA that at least covers the spatial and temporal neighborhood of the CA pattern is observed. This is the standard observability condition for all CA identification algorithms. In addition, it is assumed that the CA rule is uniform over all the observed pattern. Hybrid CAs, where more than one rule operates, cannot be dealt with by any current CA identification algorithms.

III. Identification Using the CA-OLS Method

In the present study, the problem of searching for the neighborhood and then the parameter values associated with a nonlinear logical model will initially be mapped into an equivalent polynomial representation. While this relationship is well known it has not previously been appreciated that the polynomial model can be reduced to a very simple structure with integer parameters, even for high-dimensional and complex CAs. Using this model form and introducing a modified orthogonal least squares routine, the CA-OLS method, both the CA neighborhood and the unknown polynomial model parameters can easily be determined. The equivalent polynomial model is then mapped back to a minimal logical expression to yield the final parsimonious CA model. The steps associated with this new procedure are introduced below.

A. Boolean Form of CA Rules

The local rule for a binary cellular automaton may also be considered as a Boolean function of the cells within the neighborhood. For a 1-D CA, denote the state of the cell at position j at time step t as $s(j;t)$ and the states of the cells within the neighborhood of cell j at previous time steps as $\mathbf{N}(j;|\,t)$ where $|\,t$ represents time steps before t. The 1-D CA can then be represented by

$$s(j;t) = f(\mathbf{N}(j;|\,t)) \tag{1}$$

where f is the Boolean form of the local transition rule.

Two different ways of constructing Boolean rules are currently available. One formulation produces Boolean rules using only the *NOT, AND,* and *OR* logical operators and rules for all 1-D CA with 3-site neighborhoods are listed in [8]. The Boolean form of *Rule30*, for example, is

$$\begin{aligned} s(j;t) = {} & (s(j-1;t-1) * \bar{s}(j;t-1) * \bar{s}(j+1;t-1)) \\ & \| (\bar{s}(j-1;t-1) * s(j;t-1)) \\ & \| (\bar{s}(j-1;t-1) * s(j+1;t-1)) \end{aligned} \tag{2}$$

where $\bar{}$, $*$ and $\|$ denote *NOT, AND,* and *OR* operators respectively. The alternative formulation uses only the *NOT, AND,* and *XOR* operators instead. Lists of Boolean expressions of even number 1-D 3-site CA rules based on this formulation can be found in [9]. Using these operators, *Rule30* can also be represented as

$$\begin{aligned} s(j;t) = {} & s(j-1;t-1) \oplus s(j;t-1) \\ & \oplus (\bar{s}(j;t-1) * s(j+1;t-1)) \end{aligned} \tag{3}$$

where $\oplus$ denotes the *XOR* operator. Note that on the right side of (3) logical terms are produced by connecting the "states" or the "*NOT* states" of the cells within the neighborhood using *AND* operators which are then combined by *XOR* operators. It can easily be observed that every 1-D binary rule can be reformulated into a Boolean form which follows this principle. Furthermore, note that $\bar{a} = 1 \oplus a, 0 \oplus a = a$. Hence, all 1-D binary CA can be represented by a Boolean function with only *AND* and *XOR* operators. For example, a CA with an n-site

neighborhood of $\{\text{cell}(j+1;t-1),\ldots,\text{cell}(j+n;t-1)\}$ can be expressed in the form

$$s(j;t) = a_0 \oplus a_1 s(j+1;t-1) \oplus \cdots \oplus a_N(s(j+1;t-1) * \cdots * s(j+n;t-1)) \quad (4)$$

where $a_i (i = 0,\ldots,N, N = 2^n - 1)$ are binary numbers and $a_i = 1$ indicates that the corresponding term is included in the Boolean function while $a_i = 0$ indicates that the corresponding term is not included. Note that the number of possible expressions in (4) is 2^{2^n} which is exactly the number of all possible 1-D rules with that particular neighborhood. This implies that the representation in (4) is unique, one set of $\{a_i, i = 0,\ldots,N\}$ corresponds to one and only one CA rule.

Equation (1) can be extended to higher dimensional CAs. For a three-dimensional (3-D) CA, this would be denoted as $s(i,j,l;t) = f(\mathbf{N}(i,j,l;|\,t))$, where $s(i,j,l;t)$ is the state of the cell at position (i,j,l) at time step t and $\mathbf{N}(i,j,l;|\,t)$ represents the states of the cells within the neighborhood of $\text{cell}(i,j,l;t)$. The unified expression in (4) can also be extended to multidimensional CAs. For example any 3-D CA with a 5-site neighborhood $\{\text{cell}(i-1,j,l;t-1),\text{cell}(i,j,l;t-1),\text{cell}(i,j+1,l;t-1),\text{cell}(i,j,l-1;t-1),\text{cell}(i,j,l+1;t-1)\}$ can be represented by a Boolean expression

$$s(i,j,l;t) = a_0 \oplus \cdots \oplus a_{31}(s(i-1,j,l;t-1) * \cdots * s(i,j,l+1;t-1)). \quad (5)$$

Extending this further, every CA with an n site neighborhood $\{\text{cell}(x_1;|\,t),\ldots,\text{cell}(x_n;|\,t)\}$ may be written as

$$s(x_j;t) = a_0 \oplus a_1 s(x_1;|\,t) \oplus \cdots \oplus a_N(s(x_1;|\,t) * \cdots * s(x_n;|\,t)) \quad (6)$$

where $N = 2^n - 1$ and $\text{cell}(x_j;t)$ is the cell to be updated.

Equation (6) is important because it significantly reduces the complexity of CA identification by using a reduced set of logical operators. The difficulty in identifying multidimensional CAs is also decreased because the higher dimensional CA rules are reduced to an equation which depends on the size of the neighborhood not the dimensionality.

B. Polynomial Form of CA Rules

Every CA with an n site neighborhood can be reformulated from a truth table to a Boolean function of the form of (6). However, the model to be identified is defined in terms of *AND* and *XOR* operators and is therefore nonlinear in the parameters. However, it is often advantageous to reconfigure the nonlinear model to be linear in the parameters if this is possible. This will be investigated below for CAs.

If a, a_1, a_2 are binary integer variables taking the values 0 and 1 for true and false, respectively, then there is an exact polynomial representation of each of the logical functions

$$\bar{a} = 1 - a, \quad a_1^* a_2 = a_1 \times a_2,$$
$$a_1 \oplus a_2 = a_1 + a_2 - 2a_1 \times a_2.$$

Therefore, all CA rules can be represented by exact polynomial expressions. The 1-D von Neumann *Rule30*, for example, can be written as $s(j;t) = \sum_{i=1}^{13} b_i$, where $b_1 = s(j-1;t-1); b_2 = s(j;t-1); b_3 = s(j+1;t-1); b_4 = -2s(j-1;t-1) \times s(j;t-1); b_5 = -2s(j-1;t-1) \times s(j+1;t-1); b_6 = -s(j;t-1) \times s(j+1;t-1); b_7 = 2s(j-1;t-1) \times s(j;t-1) \times s(j+1;t-1); b_8 = -2s^2(j;t-1) \times s(j+1;t-1); b_9 = -2s(j;t-1) \times s^2(j+1;t-1); b_{10} = 4s(j-1;t-1) \times s^2(j;t-1) \times s(j+1;t-1); b_{11} = 4s(j-1;t-1) \times s(j;t-1) \times s^2(j+1;t-1); b_{12} = 4s^2(j;t-1) \times s^2(j+1;t-1); b_{13} = -8s(j-1;t-1) \times s^2(j;t-1) \times s^2(j+1;t-1)$.

However, this equivalent expression will involve as many parameters as the number of possible combinations of all the cells within the neighborhood and little will be gained by using such a representation. However, using the Principle of Duality and Absorption in Boolean Algebra [10] where for every binary variable $a, a \times a = a$, considerable simplification can be achieved. Therefore, terms in the form of $s^{l_1}(j-1;t-1)s^{l_2}(j;t-1)s^{l_3}(j+1;t-1)$ where l_1, l_2, l_3 are integers can all be reduced to one term $s(j-1;t-1)s(j;t-1)s(j+1;t-1)$. Consequently, applying the Principle of Duality and Absorption to all the terms results in a new expression for all 1-D CAs with von Neumann neighborhood of the form $s(j;t) = \theta_1 s(j-1;t-1) + \theta_2 s(j;t-1) + \theta_3 s(j+1;t-1) + \theta_4 s(j-1;t-1) \times s(j;t-1) + \theta_5 s(j-1;t-1) \times s(j+1;t-1) + \theta_6 s(j;t-1) \times s(j+1;t-1) + \theta_7 s(j-1;t-1) \times s(j;t-1) \times s(j+1;t-1)$, where the parameters $\theta_1,\ldots,\theta_7$ can only take integer values and $s(j-1;t-1), s(j;t-1), s(j+1;t-1)$ are binary values. Applying this to (6) shows that a general polynomial expression of all binary CA rules with an n-site neighborhood $\{\text{cell}(x_1;|\,t),\ldots,\text{cell}(x_n;|\,t)\}$ can be expressed by the exact polynomial expression

$$s(x_j;t) = \theta_1 s(x_1;|\,t) + \cdots \theta_n s(x_n;|\,t) + \cdots + \theta_N s(x_1;|\,t) \times \cdots \times s(x_n;|\,t) \quad (7)$$

where $N = 2^n - 1$ and $\text{cell}(x_j;t)$ is the cell to be updated. Using this important observation the number of parameters to be identified can be substantially reduced to only $2^n - 1$. It can also be seen that the most important factor is the size of the neighborhood n, not the order of the dimension. For example, a 2-D CA rule with a 5-site neighborhood may have a simpler polynomial expression than a 1-D CA rule with an 8-site neighborhood. These are important observations which surprisingly have not previously been exploited and which together with the CA-OLS algorithm introduced below provide a new and powerful method of reconstructing the CA model even for high-dimensional CAs.

C. Identification Using CA-OLS

A CA can be viewed as a nonlinear dynamical system. Although the system has a spatio–temporal structure, a single time series can be measured at a single lattice site or a spatial series can be measured at a fixed time and traditional methods can be applied to model either. However, Diks [11] showed that studying only a time series or a spatial series from a spatio–temporal system without any knowledge of the system can easily lead to the incorrect conclusion that there is no spatio–temporal structure. For a full characterization of the system structure time and space have to be considered simultaneously. Determination of the spatial and temporal span of the neighborhood is therefore very important in identifying CA models.

In practice, the neighborhood structure will be unknown and it is necessary to extend the assumed neighborhood to a more general case which encompasses cells from different spatial and temporal scales. Hence a set of models which are over-specified on both the spatial and temporal spans will be introduced as the model set. For a 3-D CA, the model set can be defined as

$$\begin{aligned} s(i,j,l;t) = f(&s(i+i_1,j+j_1,l+l_1;t-1),\ldots,\\ &s(i-i_2,j-j_2,l-l_2;t-1);\ldots;\\ &s(i+i_1,j+j_1,l+l_1;t-h),\ldots,\\ &s(i-i_2,j-j_2,l-l_2;t-h)) \end{aligned} \quad (8)$$

where i_1, i_2, j_1, j_2, l_1, and l_2 denote the maximum space scale the 3-D CA could possibly span and h denotes the maximum time scale the

3-D CA could possibly span. Reducing the dimensions in (8) will yield models for 1-D and 2-D CAs as special cases while increasing the dimensions will produce models for four-dimensional (4-D), five-dimensional (5-D), and higher dimensional CAs.

Finding the neighborhood can now be defined as determining just the relevant or significant terms in (8) for the 3-D case and analogously for other dimensions. The neighborhood can therefore be thought of as equivalent to the model structure in nonlinear system identification. Using (8) and the 3-D case as an example, the neighborhood must be determined from a set of $2^{(i_1+i_2+1)(j_1+j_2+1)(l_1+l_2+1)h}-1$ possible candidate model terms. Consider, for example, a very simple 3-D CA where $i_1=i_2=j_1=j_2=l_1=l_2=h=1$; this produces $2^{27}-1=134\,217\,727$ candidate terms. This clearly shows the complexity of the task even for a simple 3-D case. Higher dimensions produce even more frightening numbers and clearly show why there are no existing solutions to these important problems. To overcome these problems the new CA-OLS algorithm is introduced below.

Initially using the 3-D case to illustrate the method, denote the states of the neighborhood $\{s(i+i_1,j+j_1,l+l_1;t-1),\ldots,s(i-i_2,j-j_2,l-l_2;t-1),\ldots,s(i+i_1,j+j_1,l+l_1;t-h),\ldots,s(i-i_2,j-j_2,l-l_2;t-h)\}$ in (8) as $\{u_1,\ldots,u_n\}$, where the size of the neighborhood $n=(j_1+j_2+1)(i_1+i_2+1)(l_1+l_2+1)h$. Then expanding (8) into the polynomial form shown in (6) yields

$$s(i,j,l;t)=\theta_1u_1+\cdots+\theta_nu_n+\cdots+\theta_Nu_1\times\cdots\times u_n \qquad (9)$$

where $N=2^n-1$, and $\theta_1,\ldots,\theta_N$ are a set of integers such that (9) maps $s(i,j,l;t)$ into $\{0,1\}$. Note that (9) can be readily extended from the 3-D case to be valid for all binary CAs.

The CA-OLS algorithm is derived by applying a modified Gram–Schmidt orthogonal procedure to (9). The CA-OLS algorithm is given in the Appendix.

The simple 3-D example above shows the number of possible candidate terms can be excessive, but simulations by many authors show that often complex CA patterns can be produced using simple models. If the appropriate terms that are significant can be selected therefore the remainder can be discarded without any deterioration in model precision or prediction accuracy and a concise CA model can be obtained. One way to determine which terms are significant or which should be included in the model can be derived as a by-product of the CA-OLS estimation algorithm and is very simple to implement. From the Appendix, the quantity $[ct]$ is defined as

$$[ct]_d=\frac{\tilde{\theta}_d^2\times\sum_{t=1}^{M}e_d^2(t)}{\sum_{t=1}^{M}s^2(i,j,l;t)}$$

and measures the contribution that each candidate term makes to the updated state $s(i,j,l;t)$ and provides an indication of which terms to include in the model. Using $[ct]$ the candidate model terms can be ranked in order of importance and insignificant terms can be discarded by defining a value of $[ct]$, below which terms are considered to contribute a negligible reduction in the mean-squared error. The threshold value of $[ct]$ for the CA model can be set to zero because the polynomial model is not an approximation but an exact representation of the CA rules. The threshold value is set to zero to ensure that sufficient terms are included and the prediction errors are reduced to zero. Notice that the forward-regression orthogonal algorithm [12] is used in the Appendix, this provides a $[ct]$ test which is independent of the order of inclusion of terms in the model. The structure of the neighborhood is therefore defined by retaining only the significant $[ct]$ terms and the CA rule can then be computed by linearly combining all the selected terms with the estimated parameters.

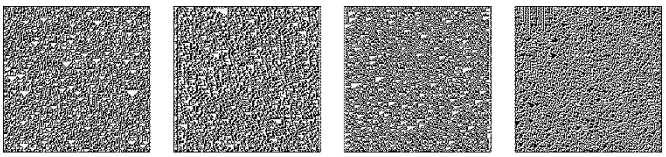

Fig. 2. Evolution of the 1-D CA *Rule30* with four different neighborhoods: (a) *Rule30* (01 111 000) von Neumann; (b) *Rule30* left-shift; (c) *Rule30* right-shift; and (d) *Rule30* temporal-shift.

D. Extracting the Boolean Form of the CA Rules

The polynomial form of the model can be determined using the orthogonal estimator which yields both the CA neighborhood and the model parameters. Although the polynomial model can be used to directly reproduce the complex spatio–temporal patterns, hardware realization of the CA may not be straightforward based on the polynomial form and it is therefore important to extract the equivalent Boolean rules. While it is straightforward to extract canonical forms [10] of Boolean functions from truth tables constructed on the basis of polynomial rules the canonical forms are often unwieldy and typically more operations than are necessary are involved. However, this problem can be solved by using the Quine–McCluskey [10] method to extract the parsimonious Boolean expressions from identified polynomials. The Boolean rules extracted using the Quine–McCluskey method involves *NOT, AND,* and *OR* operators. To obtain rules employing *NOT, AND,* and *XOR* instead, see details in [9, Ch. 1].

IV. Simulation Studies

Three simulation examples are included to demonstrate the application of the new algorithm. Initially a simple 1-D example will be discussed to show all the steps involved in a transparent manner. More realistic 2-D and 4-D examples will then be discussed.

A. Identification of 1-D 3-Site CA Rule30

The spatio–temporal patterns generated by the evolution of *Rule30* on a 200×200 lattice with four different neighborhoods, a von Neumann neighborhood $\{\text{cell}(j-1;t-1),\text{cell}(j;t-1),\text{cell}(j+1;t-1)\}$, a left-shift neighborhood $\{\text{cell}(j-2;t-1),\text{cell}(j-1;t-1),\text{cell}(j;t-1)\}$, a right-shift neighborhood $\{\text{cell}(j;t-1),\text{cell}(j+1;t-1),\text{cell}(j+2;t-1)\}$, and a temporal-shift neighborhood $\{\text{cell}(j-1;t-2),\text{cell}(j;t-1),\text{cell}(j+1;t-1)\}$ are shown in Fig. 2(a), (b), (c), and (d), respectively. An initial inspection of Fig. 2(a), (b), (c), and (d) shows that the structure of the neighborhood corresponds to the pattern produced. The randomly distributed triangle structures in Fig. 2(b) are simply the left half of the triangles in Fig. 2(a), whereas Fig. 2(c) is composed of the right half of the triangles in Fig. 2(a). The patterns demonstrate the difference among these three neighborhoods. The pattern in Fig. 2(d) is produced by operating *Rule30* on a temporal-shift neighborhood which involves cells from both time steps $t-1$ and $t-2$. However, the blurred image and the rotated triangles that this produces barely shows any resemblance to the patterns in Fig. 2(a), (b), or (c).

Because of the special construction of CA which are synchronously updated using the same Boolean function over the whole lattice, the data points that are available are redundant for identification purposes and can be extracted in two ways as shown in Fig. 3(a) and (b), respectively. In Fig. 3(a), data points are extracted row by row/spacewise, while in Fig. 3(b), data points are extracted column by column/timewise. Since each cell is synchronously updated under the same Boolean function, a change of the rows or columns when extracting the data is

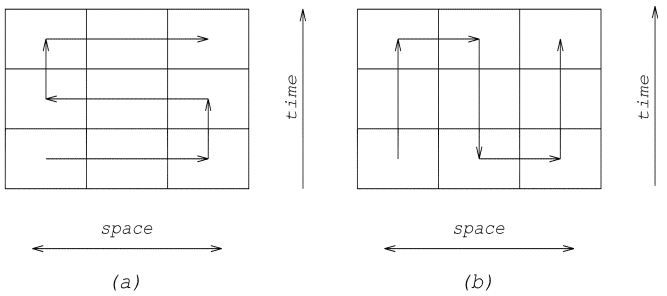

Fig. 3. Extracting data points from 1-D CA patterns.

not important. Assume initially that the largest possible neighborhood is defined by $\{\text{cell}(j-2;t-1), \text{cell}(j-1;t-1), \text{cell}(j;t-1), \text{cell}(j+1;t-1), \text{cell}(j+2;t-1), \text{cell}(j+1;t-2)\}$ and hence define the neighborhood vector

$$a(t) = [s(j-2;t-1)s(j-1;t-1)s(j;t-1) \times s(j+1;t-1)s(j+2;t-1)s(j+1;t-2)]^T. \quad (10)$$

The candidate model term set (MT) will initially be constructed as

$$\text{MT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ & & \vdots & & & \\ 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ & & \vdots & & & \\ 5 & 6 & 0 & 0 & 0 & 0 \\ & & \vdots & & & \\ 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

where 1, 2, 3, 4, 5, and 6 denote the rows in the neighborhood vector. For example, an entry of "4'" represents the fourth row in (10) and is therefore associated with $\text{cell}(j+1;t-1)$, and so on. The full model set MT consists of $N = 63$ terms/rows. Each row in this model represents a candidate term which corresponds to a term $s_d, d = 1, \ldots, N$ in (11) in the Appendix. For example, the first row (1 0 0 0 0 0) represents $s(j-2;t-1)$ only while the last row (1 2 3 4 5 6) corresponds to a product of six states $s(j-2;t-1) \times s(j-1;t-1) \times s(j;t-1) \times s(j+1;t-1) \times s(j+2;t-1) \times s(j+1;t-2)$.

Five hundred data points were extracted from the patterns in Fig. 2(a), (b), (c), and (d), respectively, and these were used to fit the models. No *a priori* information regarding the neighborhoods or rules was assumed. In the simulation the threshold for the term contribution $[ct]$ values for the four models were all chosen as 0 and the CA-OLS estimator searched through 63 possible candidate terms for each model. Finally, four different models were selected which were associated with four different neighborhoods. The models and the corresponding parameters θ are shown in models 1-(*a*), 1-(*b*), 1-(*c*), and 1-(*d*).

$$\text{MT} = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 4 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \theta = \begin{bmatrix} 1.0000 \\ -2.0000 \\ 1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \end{bmatrix}$$

Model 1-(*a*)

$$\text{MT} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \theta = \begin{bmatrix} 1.0000 \\ -2.0000 \\ 1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \end{bmatrix}$$

Model 1-(*b*)

$$\text{MT} = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \theta = \begin{bmatrix} 1.0000 \\ -2.0000 \\ 1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \end{bmatrix}$$

Model 1-(*c*)

$$\text{MT} = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 \\ 3 & 6 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 3 & 4 & 6 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \theta = \begin{bmatrix} 1.0000 \\ -2.0000 \\ 1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \end{bmatrix}$$

Model 1-(*d*)

The terms in model 1-(*b*) represent the left-shift neighborhood because all the model entries are selected from $\{\text{cell}(j-2;t-1), \text{cell}(j-1;t-1),$ and $\text{cell}(j;t-1)\}$. Notice how the CA-OLS algorithm has correctly selected only the appropriate three cells and discarded the remainder. Combining the terms with the corresponding parameters θ, the identified model describes the CA rule in the polynomial form: $s(j;t) = s(j-1;t-1) - 2s(j-2;t-1) \times s(j-1;t-1) + s(j-2;t-1) - 2s(j-2;t-1) \times s(j;t-1) + s(j;t-1) + 2s(j-2;t-1) \times s(j-1;t-1) \times s(j;t-1) - s(j-1;t-1) \times s(j;t-1)$. The terms in model 1-(*a*) represent the von Neumann neighborhood $\{\text{cell}(j-1;t-1), \text{cell}(j;t-1), \text{cell}(j+1;t-1)\}$ while the terms in model 1-(*c*) correspond to the right-shift neighborhood $\{\text{cell}(j;t-1), \text{cell}(j+1;t-1), \text{cell}(j+2;t-1)\}$. The result in model 1-(*d*) covers entry 6, which in (10) represents a cell at time step $t-2, \text{cell}(j-1;t-2)$. Model 1-(*d*) therefore defines a temporal-shift neighborhood involving $\text{cell}(j-1;t-2), \text{cell}(j,t-1)$, and $\text{cell}(j+1,t-1)$.

In each case, the CA-OLS algorithm has correctly determined the appropriate neighborhood. Notice that the parameters θ in models 1-(*a*), 1-(*b*), 1-(*c*), and 1-(*d*) are all exactly the same but that each operates on a different neighborhood and hence produces a different CA pattern. The measured output and the model predicted output (MPO) for the von Neumann neighborhood are compared in Fig. 4 where the MPO is defined as

$$\hat{s}_m(j;t) = \hat{f}(\hat{s}_m(j+j_1;t-1), \ldots, \hat{s}_m(j-j_2;t-1); \ldots; \hat{s}_m(j+j_1;t-h), \ldots, \hat{s}_m(j-j_2;t-h)).$$

The MPO is a more strict criteria for evaluating the performance of the estimator than the one-step ahead prediction (OSA) which is defined as

$$\hat{s}_m(j;t) = \hat{f}(s_m(j+j_1;t-1), \ldots, s_m(j-j_2;t-1); \ldots; s_m(j+j_1;t-h), \ldots, s_m(j-j_2;t-h)).$$

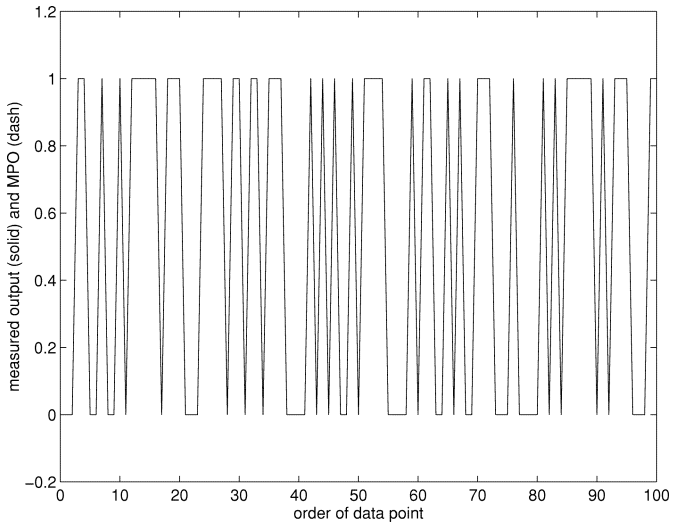

Fig. 4. Comparison of the measured (solid line) and the model predicted output (dashed line) for *Rule30* with a von Neumann neighborhood.

The comparison in Fig. 4 clearly shows that the measured output and the predicted output using the estimated model are almost coincidental. The dashed line follows the solid line without any deviation. The variance is virtually zero because the polynomial expression is not an approximation of the CA rules but an equivalent representation. The MPOs for other neighborhoods also match the corresponding measured outputs exactly. For simplicity, the comparisons are not shown in this paper. The Quine–McCluskey [10] method was then used to extract the minimum Boolean expression from the estimated polynomial form for *Rule30* with the left-shift neighborhood. The Boolean expression obtained was $s(j;t) = \bar{s}(j-2;t-1) * s(j;t-1) \oplus \bar{s}(j-2;t-1) * s(j-1;t-1) \oplus s(j-2;t-1) * \bar{s}(j-1;t-1) * \bar{s}(j;t-1)$ which corresponds exactly to the entry of the Boolean expression for *Rule30* in [8]. Similarly, applying the Quine–McCluskey method to models 1-(*a*), 1-(*c*), and 1-(*d*) will produce correct results, respectively.

B. Identification of a 2-D 5-Site CA Rule

The spatio–temporal patterns produced by a 2-D CA evolution on a 200×200 lattice with a von Neumann neighborhood are illustrated in Fig. 5. Assume initially that the largest possible neighborhood for this 2-D rule is the 9-site Moore neighborhood. Define the neighborhood vector as

$$\begin{aligned} a(t) = [&s(i-1,j-1;t-1) \\ &\times s(i-1,j;t-1)s(i-1,j+1;t-1) \\ &\times s(i,j-1;t-1)s(i,j+1;t-1)s(i+1,j-1;t-1) \\ &\times s(i+1,j;t-1)s(i+1,j+1;t-1)s(i,j;t-1)]^T. \end{aligned}$$

The initial model was constructed as

$$\mathrm{MT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & \vdots & & & & \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

where 1, 2, 3, 4, 5, 6, 7, 8, and 9 represent $\mathrm{cell}(i-1,j-1;t-1)$, $\mathrm{cell}(i-1,j;t-1)$, $\mathrm{cell}(i-1,j+1;t-1)$, $\mathrm{cell}(i,j-1;t-1)$, $\mathrm{cell}(i,j+1;t-1)$, $\mathrm{cell}(i+1,j-1;t-1)$, $\mathrm{cell}(i+1,j;t-1)$, $\mathrm{cell}(i+1,j+1;t-1)$, and $\mathrm{cell}(i,j;t-1)$, respectively.

$$\mathrm{MT} = \begin{bmatrix} 4 & 7 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 7 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 5 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 5 & 9 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \theta = \begin{bmatrix} -1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ -2.0000 \\ -1.0000 \\ 1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \\ 2.0000 \\ -2.0000 \\ -1.0000 \\ -1.0000 \\ 1.0000 \end{bmatrix}$$

Model 2-(a)

One thousand data points were extracted from the CA patterns in Fig. 5 and the threshold for the $[ct]$ cutoff was set to zero. The CA-OLS estimator produced a model with only 17 rows after searching through the whole model set of $N = 2^9 - 1 = 511$ candidate terms. The identified model 2-(*a*) clearly shows that the CA-OLS algorithm has correctly selected cells 2, 4, 5, 7, and 9 which correspond to the von Neumann neighborhood $\{\mathrm{cell}(i-1,j;t-1), \mathrm{cell}(i,j-1;t-1), \mathrm{cell}(i,j;t-1), \mathrm{cell}(i,j+1;t-1),$ and $\mathrm{cell}(i+1,j;t-1)\}$, respectively. The MPO was exactly equal to the measured output over the data points. For simplicity, this comparison is not shown in this paper.

Applying the Quine–McCluskey method to model 2-(*a*), the following final prime implicants were obtained:

$$\begin{aligned} &A{:}\bar{s}(i,j;t-1) * \bar{s}(i,j+1;t-1) * s(i-1,j;t-1) \\ &B{:}\bar{s}(i,j-1;t-1) * s(i,j+1;t-1) * \bar{s}(i-1,j;t-1) \\ &C{:}\bar{s}(i,j;t-1) * s(i,j+1;t-1) * \bar{s}(i-1,j;t-1) \\ &D{:}\bar{s}(i+1,j;t-1) * s(i,j;t-1) * \bar{s}(i-1,j;t-1) \\ &E{:}\bar{s}(i,j-1;t-1) * s(i,j;t-1) * \bar{s}(i,j+1;t-1) \\ &F{:}\bar{s}(i+1,j;t-1) * s(i,j-1;t-1) * \bar{s}(i,j+1;t-1) \\ &G{:}s(i,j-1;t-1) * \bar{s}(i,j;t-1) * \bar{s}(i,j+1;t-1) \\ &H{:}\bar{s}(i+1,j;t-1) * s(i,j-1;t-1) * \bar{s}(i-1,j;t-1) \\ &I{:}s(i,j-1;t-1) * \bar{s}(i,j;t-1) * \bar{s}(i-1,j;t-1) \\ &J{:}s(i,j-1;t-1) * \bar{s}(i,j;t-1) * \bar{s}(i,j+1;t-1) \\ &K{:}s(i,j-1;t-1) * \bar{s}(i,j+1;t-1) * s(i-1,j;t-1) \\ &L{:}s(i+1,j;t-1) * \bar{s}(i,j+1;t-1) * s(i-1,j;t-1) \\ &M{:}s(i+1,j;t-1) * \bar{s}(i,j-1;t-1) * s(i,j;t-1). \end{aligned}$$

Finally, the Boolean expression of this 5-site 2-D rule is the *OR* combination of all the above A–M 13 items.

C. Identification of a 4-D CA

Data extracted from a 4-D $12 \times 12 \times 12 \times 12$ cellular automaton with null boundary conditions will be used to illustrate the identification. The initial neighborhood was assumed to encompass $\{\mathrm{cell}(i-1,j,l,k;t-1), \mathrm{cell}(i+1,j,l,k;t-1), \mathrm{cell}(i,j-$

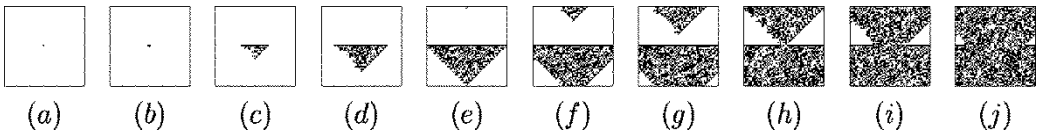

Fig. 5. Evolution of a 2-D CA rule with a 5-site von Neumann neighborhood: (a) $t = 0$; (b) $t = 1$; (c) $t = 10$; (d) $t = 20$; (e) $t = 30$; (f) $t = 40$; (g) $t = 50$; (h) $t = 60$; (i) $t = 70$; and (j) $t = 80$.

$1, l, k; t-1), \text{cell}(i, j+1, l, k; t-1), \text{cell}(i, j, l-1, k; t-1), \text{cell}(i, j, l+1, k; t-1), \text{cell}(i, j, l, k-1; t-1), \text{cell}(i, j, l, k+1; t-1), \text{cell}(i, j, l, k; t-1)\}$. This, in turn, defines the neighborhood vector as $a(t) = [s(i-1, j, l, k; t-1)s(i+1, j, l, k; t-1)s(i, j-1, l, k; t-1)s(i, j+1, l, k; t-1)s(i, j, l-1, k; t-1)s(i, j, l+1, k; t-1)s(i, j, l, k-1; t-1)s(i, j, l, k+1; t-1)s(i, j, l, k; t-1)]$. The initial model was constructed as

$$\text{MT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & \vdots & & & & \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

where 1, 2, 3, 4, 5, 6, 7, 8, and 9 represent $\{\text{cell}(i-1, j, l, k; t-1), \text{cell}(i+1, j, l, k; t-1), \text{cell}(i, j-1, l, k; t-1), \text{cell}(i, j+1, l, k; t-1), \text{cell}(i, j, l-1, k; t-1), \text{cell}(i, j, l+1, k; t-1), \text{cell}(i, j, l, k-1; t-1), \text{cell}(i, j, l, k+1; t-1),$ and $\text{cell}(i, j, l, k; t-1)\}$, respectively. The $[ct]$ threshold was again set to zero and the number of possible candidate models was $2^9 - 1 = 511$. The CA-OLS estimator produced a model with only 14 rows and the associated integer parameters given in model 4-(a)

$$\text{MT} = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 5 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \theta = \begin{bmatrix} 1.0000 \\ 1.0000 \\ 1.0000 \\ -1.0000 \\ -1.0000 \\ 1.0000 \\ -1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ 1.0000 \end{bmatrix}$$

Model 4-(a).

A comparison of the measured output and MPO again produced virtually coincidental results showing that the correct model of the CA has been estimated. For simplicity, this comparison is not shown in this paper. Comparing the estimated model 4-(a) with model 2-(a) shows that the structure in model 4-(a) is much simpler than that in model 2-(a) although the former is extracted from a 4-D CA while the later is from a 2-D CA. The computation time for both are approximately the same. It can be seen that the efficiency of the CA-OLS estimator relies largely on the size of the neighborhood, that is the number of cells within the neighborhood, rather than the order or dimension of the CA and this can dramatically simplify the problem of identifying higher dimensional CA.

Applying the Quine–McCluskey method to the polynomial produced from model 4-(a) the correct Boolean expression of the *OR* combination of nine prime implicants was obtained. For simplicity, the prime implicants are not listed.

V. Conclusion

While many authors have demonstrated that simple CA models can produce complex spatio–temporal patterns, few investigators have studied how to recover such models given only the data patterns. One possible solution to this important problem has been introduced in this study using the new CA-OLS estimator.

The new estimator exploits the observation that binary CA rules can be exactly represented as polynomial models which collapse to relatively simple forms even for high-dimensional CAs. This transforms the problem from a nonlinear-in-the-parameters to a linear-in-the -parameters formulation. The neighborhood of the CA can then be determined using a modified orthogonal least squares estimator. Identifying the neighborhood of the CA is critical if the underlying rules are to be estimated and it has been shown that the term contribution test is an efficient solution to this problem. Once the neighborhood and the polynomial model parameters have been obtained, the model can then be mapped back to a Boolean form using the Quine–McCluskey method.

The only information required is to set the range of the largest expected neighborhood over which the algorithm searches for candidate model terms. The CA-OLS estimator then searches through all the possible terms and discards all terms below the $[ct]$ threshold to yield the estimated model. The MPO is used as a metric of performance to validate the model.

Several simulated examples show the power of the new approach and demonstrate for the first time how CA models can be extracted from data generated from high-dimensional CA systems.

Further research is required to address the case of hybrid CAs, where more than one CA rule applies in a pattern, and to confirm that the CA-OLS algorithm can be used to identify n-dimensional CA rule given the evolution over an $n+1$-dimensional space.

Appendix

A. The CA-OLS Algorithm

Consider the polynomial expression for 3-D CAs in (9), for example. Denote $(u_1, \ldots, u_n, \ldots, u_1 \times \cdots \times u_n)$ as $(s_1, \ldots, s_N)$. Equation (9) can then be written as

$$s(i, j, l; \tau) = \sum_{d=1}^{N} s_d(\tau) \times \theta_d = \mathbf{s}(\tau) \times \bar{\theta} \tag{11}$$

where τ indicates the order of the data point and

$$\bar{\theta} = [\theta_1 \quad \theta_2 \quad \cdots \quad \theta_N]^T$$
$$\mathbf{s}(\tau) = [s_1(\tau) \quad s_2(\tau) \quad \cdots \quad s_N(\tau)].$$

Equation (11) can also be represented in a matrix form as

$$\mathbf{s}(i, j, l) = \mathbf{S} \times \bar{\theta} \tag{12}$$

where

$$\mathbf{s}(i, j, l) = [s(i, j, l; 1) \quad s(i, j, l; 2) \quad \cdots \quad s(i, j, l; M)]^T$$
$$\mathbf{S} = [\mathbf{s}^T(1) \quad \mathbf{s}^T(2) \quad \cdots \quad \mathbf{s}^T(M)]^T = [\mathbf{s}_1 \quad \cdots \quad \mathbf{s}_N]$$

and M denotes the number of data points in the data set. Matrix $\mathbf{S}$ can be decomposed as $\mathbf{S} = \mathbf{E} \times \mathbf{Q}$, where

$$\mathbf{E} = \begin{bmatrix} e_1(1) & \cdots & e_N(1) \\ \vdots & & \vdots \\ e_1(M) & \cdots & e_N(M) \end{bmatrix} = [\mathbf{e}_1 \quad \cdots \quad \mathbf{e}_N]$$

is an orthogonal matrix, that is, $\mathbf{E}^T \times \mathbf{E} = \mathrm{Diag}[\mathbf{e}_1^T \times \mathbf{e}_1 \ \cdots \ \mathbf{e}_N^T \times \mathbf{e}_N]$ and $\mathbf{Q}$ is an upper triangular matrix with unity diagonal elements

$$\mathbf{Q} = \begin{bmatrix} 1 & q_{12} & q_{13} & \cdots & q_{1N} \\ & 1 & q_{23} & \cdots & q_{2N} \\ & & \ddots & \ddots & \vdots \\ & & & 1 & q_{N-1N} \\ & & & & 1 \end{bmatrix}.$$

Equation (12) can then be represented as

$$\mathbf{s}(i,j,l) = \mathbf{E} \times \mathbf{Q} \times \bar{\theta} = \mathbf{E} \times \bar{\tilde{\theta}} \tag{13}$$

where $\bar{\tilde{\theta}} = \mathbf{Q} \times \bar{\theta} = [\tilde{\theta}_1 \ \cdots \ \tilde{\theta}_N]^T$.

Therefore, (11) can be written as

$$s(i,j,l;\tau) = \sum_{d=1}^{N} e_d(\tau) \times \tilde{\theta}_d. \tag{14}$$

The contribution each term $\{\mathbf{s}_d, d = 1, \ldots, N\}$ in (12) makes to $\mathbf{s}(i,j,l)$ can then be calculated as

$$[ct]_d = \frac{\tilde{\theta}_d^2 \times \sum_{\tau=1}^{M} e_d^2(\tau)}{\sum_{\tau=1}^{M} s^2(i,j,l;\tau)}. \tag{15}$$

The sum of all the $[ct]$ values will be unity so if $[ct]$ were multiplied by 100 this would give the percentage contribution that each term makes to $\mathbf{s}(i,j,l)$. The orthogonalization of $\mathbf{S}$ simplifies the term selection process and allows each relevant term to be added to the identified term set MT independently of other terms. The parameter vector $\bar{\theta}$ can then be estimated by computing each $\tilde{\theta}_d$ one at a time. However, in the term selection process, $[ct]_d$ may depend on the order in which $s_d(\tau)$ enters (11). A change of the position of $s_d(\tau)$ in (11) may result in a change of the associated $[ct]_d$ value. Consequently, simply orthogonalizing the columns in $\mathbf{S}$ into (13) in the order in which $s_d(\tau)$s happen to appear in (11) may produce the wrong information regarding the corresponding contributions. To avoid this problem, the following forward regression algorithm is used. This algorithm will forward add terms instead of forward deleting terms and will therefore disregard the order that $s_d(\tau)$ enters (11).

The forward regression CA-OLS algorithm is given by the following.

1) Consider all the $s_d(\tau)$ as possible candidates for $e_1(\tau)$. For $d = 1, \ldots, N$, calculate

$$e_1^{(d)}(\tau) = s_d(\tau)$$
$$\tilde{\theta}_1^{(d)} = \frac{\sum_{\tau=1}^{M} e_1^{(d)}(\tau) s(i,j,l;\tau)}{\sum_{\tau=1}^{M} \left(e_1^{(d)}(\tau)\right)^2}$$
$$[ct]_1^{(d)} = \frac{\left(\tilde{\theta}_1^{(d)}\right)^2 \sum_{\tau=1}^{M} \left(e_1^{(d)}(\tau)\right)^2}{\sum_{\tau=1}^{M} s^2(i,j,l;\tau)}.$$

Find and denote the maximum of $[ct]_1^{(d)}$ as $[ct]_1^{(v)} = \max\{[ct]_1^{(d)}, 1 \le d \le N\}$. The first relevant term $e_1(\tau)$ is selected as $e_1^{(v)}(\tau)$ and $\tilde{\theta}_1 = \tilde{\theta}^{(v)}, [ct]_1 = [ct]_1^{(v)}$. The corresponding $s_v(\tau)$ is then included in the identified model set MT.

2) All of the $s_d(\tau), d = 1, \ldots, N, d \neq v$ are considered as possible candidates for $e_2^{(v)}(\tau)$. For $d = 1, \ldots, N, d \neq v$, calculate

$$e_2^{(d)}(\tau) = s_d(\tau) - q_{12}^{(d)} e_1(\tau)$$
$$\tilde{\theta}_2^{(d)} = \frac{\sum_{\tau=1}^{M} e_2^{(d)}(\tau) s(i,j,l;\tau)}{\sum_{\tau=1}^{M} \left(e_2^{(d)}(\tau)\right)^2}$$
$$[ct]_2^{(d)} = \frac{\left(\tilde{\theta}_2^{(d)}\right)^2 \sum_{\tau=1}^{M} \left(e_2^{(d)}(\tau)\right)^2}{\sum_{\tau=1}^{M} s^2(i,j,l;\tau)}$$

where

$$q_{12}^{(d)} = \frac{\sum_{\tau=1}^{M} e_1(\tau) s_d(\tau)}{\sum_{\tau=1}^{M} e_1^2(\tau)}.$$

Find and denote the maximum of $[ct]_2^{(d)}$ as $[ct]_2^{(g)} = \max\{[ct]_2^{(d)}, 1 \le d \le N, d \neq v\}$. The second term $e_2(\tau)$ is therefore selected as $e_2^{(g)}(\tau) = s_g(\tau) - q_{12}^{(g)} e_1(\tau)$ and $q_{12} = q_{12}^{(g)}, \tilde{\theta}_2 = \tilde{\theta}^{(g)}, [ct]_2 = [ct]_2^{(g)}$. The corresponding $s_g(\tau)$ is then included in the identified model set MT.

3) The procedure is terminated at the N_sth step either when $1 - \sum_{d=1}^{N_s} [ct]_d < C_{\text{off}}$ (desired tolerance), $N_s < N$, or when $N_s = N$.

4) From the selected orthogonal equation $s(i,j,l;\tau) = \sum_{d=1}^{N_s} e_d(\tau) \tilde{\theta}_d$, it is then straightforward to calculate the corresponding N_s parameters θ using $\theta_{N_s} = \tilde{\theta}_{N_s}, \theta_m = \tilde{\theta}_m - \sum_{k=m+1}^{N_s} q_{mk} \theta_k, m = N_s - 1, \ldots, 1$.

REFERENCES

[1] G. Hernandez and H. J. Herrann, "Cellular automata for elementary image enhancement," *Graph. Models Image Process.*, vol. 58, no. 1, pp. 82–89, 1996.

[2] P. Tzionas *et al.*, "Design and vlsi implementation of a pattern classifier using pseudo 2d cellular automata," in *Proc. Inst. Elect. Eng., Circuits, Devices and Systems*, vol. 139, 1992, pp. 661–668.

[3] E. Macii and M. Poncino, "Cellular automata models for reliablity analysis of systems on silicon," *IEEE Trans. Rel.*, vol. 146, no. 2, pp. 173–183, 1997.

[4] S. Surka and K. P. Valavanis, "A cellular automata model for edge relaxation," *J. Intell. Robot. Syst.*, vol. 4, no. 4, pp. 379–391, 1991.

[5] A. I. Adamatzky, *Identification of Cellular Automata*. New York: Taylor & Francis, 1994.

[6] F. C. Richards, "Extracting cellular automaton rules directly from experimental data," *Phys. D*, vol. 45, pp. 189–202, 1990.

[7] Y. X. Yang and S. A. Billings, "Extracting Boolean rules from CA patterns," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, pp. 573–580, Aug. 2000.

[8] S. Wolfram, *Cellular Automata and Complexity*, ser. MA. Reading: Addison-Wesley, 1994.

[9] B. H. Voorhees, *Computational Analysis of One-Dimensional Cellular Automata, World Scientific Series on Nonlinear Science*, Singapore: World Scientific, 1996.

[10] R. Korfhage, *Logic and Algorithms*. New York: Wiley, 1996.

[11] C. Diks *et al.*, "Spatio–temporal chaos: A solvable model," *Phys. D*, vol. 104, pp. 269–285, 1997.

[12] S. A. Billings, S. J. Chen, and M. J. Korenberg, "Identification of mimo nonlinear systems using a forward-regression orthogonal estimator," *Int. J. Contr.*, vol. 49, no. 6, pp. 2157–2189, 1989.