Parallel Computing 23:11 (1997) 1699-1715

Cellular Automata for Reaction-Diffusion Systems

Jörg R. Weimar
Institute for Scientific Computing, Technical University Braunschweig
D–38092 Braunschweig, Germany
Tel. +49-531-191-3006; E-mail J.Weimar@tu-bs.de

December 6, 1996

Abstract

A class of cellular automata for reaction-diffusion systems is presented. It is based on a local average for the diffusive dynamics, and closely related to finite difference schemes. The reactive dynamics is implemented as a lookup-table with probabilistic rules. The rules are derived directly and systematically from the given differential equations, using probabilistic rounding to enforce the discretization of the concentration variables. For quantitatively correct modeling, such probabilistic rules are usually necessary, but in some cases a deterministic version proves sufficient.

Keywords: Cellular automata, reaction-diffusion systems, probabilistic rounding, simulation, modeling.

Introduction

Reaction-diffusion systems are an important class of systems to investigate nonlinear behavior. They also represent many problems arising in chemistry, biology, and other disciplines. Nonlinear reaction-diffusion systems can be simulated by standard numerical techniques, such as finite difference methods or using alternative approaches such as cellular automata. Many cellular automata (CA) for reaction-diffusion systems are constructed in such a way that they correspond qualitatively to the solutions of the partial differential equations [3, 6]. Here we introduce a class of CA that models the partial differential equations also in a quantitatively correct way. Other CA that can be compared quantitatively to solutions of the partial differential equations (PDEs) include CA for excitable media [5, 11] and the class of reactive lattice gas automata [1, 2, 9]. The former are restricted to certain phenomena and parameters and the latter are most useful in situations where microscopic fluctuations play an important role. When fluctuations are not important, a more macroscopic approach as presented here is more efficient. The CA presented here is always more efficient than explicit numerical techniques, and can in many cases be more efficient than better numerical techniques.

The equation to be simulated is

$$\frac{\partial x(\mathbf{r},t)}{\partial t} = D\nabla^2 x(\mathbf{r},t) + f(x(\mathbf{r},t)), \tag{1}$$

where $\mathbf{r}$ gives the position in the spatial domain and $f(x)$ is a nonlinear function, the rate law. In general, x can also have more than one component. In this case the Laplacian operates on each component independently, but the nonlinear function f couples the different components.

We present here a systematic construction for a class of CA to simulate such equations. Each CA step can be decomposed into two parts, one to implement the diffusion ($D\nabla^2 x(\mathbf{r}, t)$) and the other to implement the reaction ($f(x)$) and the rounding necessitated by the use of a finite set of states.

Moving Average for Diffusion

For the diffusive part, we use a modified finite difference scheme (see also [9, 10]), while for the reactive part we use probabilistic rules.

The one-dimensional diffusion equation

$$\frac{\partial x(r,t)}{\partial t} = D\nabla^2 x(r,t) \tag{2}$$

is usually discretized as

$$\frac{x(r,t+\Delta t) - x(r,t)}{\Delta t} = D\frac{x(r-\Delta r,t) - 2x(r,t) + x(r+\Delta r,t)}{\Delta r^2}, \tag{3}$$

which can be rewritten as

$$x(r,t+\Delta t) \quad = \quad D\frac{\Delta t}{\Delta r^2}x(r-\Delta r,t) + \left(1 - 2D\frac{\Delta t}{\Delta r^2}\right)x(r,t) + D\frac{\Delta t}{\Delta r^2}x(r+\Delta r,t). \tag{4}$$

We can generalize this discretization and use more general coefficients a_i:

$$x(r,t+\Delta t) = \sum_{i=-R}^{R} a_i\, x(r + i\,\Delta r, t). \tag{5}$$

By using a Taylor-expansion in space for the $x(r + i\Delta r)$, we obtain

$$x(r,t+\Delta t) = c_0 x(r,t) + c_1\frac{\partial x(r,t)}{\partial r} + \frac{c_2}{2}\frac{\partial^2 x(r,t)}{\partial r^2} + \mathrm{O}(\Delta r^3) \tag{6}$$

with

$$c_0 \quad = \quad \sum_{i=-R}^{R} a_i \tag{7a}$$

$$c_1 \quad = \quad \sum_{i=-R}^{R} (i\Delta r)a_i \tag{7b}$$

$$c_2 \quad = \quad \sum_{i=-R}^{R} (i\Delta r)^2 a_i. \tag{7c}$$

We obtain an approximation to the diffusion equation if we also Taylor-expand $x(r, t + \Delta t)$ in t and set

$$c_0 \quad = \quad 1 \tag{8a}$$

$$c_1 \quad = \quad 0 \tag{8b}$$

$$c_2 \quad = \quad 2\,\Delta t\; D. \tag{8c}$$

Eq. (8a) is a normalization condition. For $c_1 \neq 0$, there would be an additional drift term, and eq. (8c) simply specifies the effective diffusion coefficient. A special case is the situation where all the a_i are equal. Then $a = a_i = \frac{1}{2R+1}$ from eq. (8a) and $D = \frac{R(R+1)}{6} \frac{\Delta r^2}{\Delta t}$ from eq. (8c). This choice of coefficients is particularly useful since the calculation of the sum in eq. (5) can be performed as a moving average: If

$$x(r, t + \Delta t) = a \sum_{i=-R}^{R} x(r + i\Delta r, t). \tag{9}$$

then

$$x(r + \Delta r, t + \Delta t) \;=\; x(r, t + \Delta t) + a \left(x(r + (R+1)\Delta r, t) - x(r - R\Delta r, t) \right), \tag{10}$$

and proceeding thus along the r-axis, we can calculate the new value $x(r, t+\Delta t)$ for each r with only two additions per cell instead of $(2R+1)$ additions.

This method is easily generalized to two (or more) dimensions: In this case $\mathbf{r}$ is a vector and

$$x(\mathbf{r}, t + \Delta t) = \sum_{i=-R}^{R} \sum_{j=-R}^{R} a_{i,j}\; x\left(\mathbf{r} + (i,j)^T \Delta r, t\right), \tag{11}$$

$a_{i,j} = a = \frac{1}{(2R+1)^2}$, and $D = \frac{R(R+1)}{6} \frac{\Delta r^2}{\Delta t}$. We perform the calculations

$$\begin{aligned}
\tilde{x}\left(\mathbf{r} + (1,0)^T \Delta r, t\right) &= \tilde{x}(\mathbf{r}, t) + x\left(\mathbf{r} + (R+1,0)^T \Delta r, t\right) - x\left(\mathbf{r} - (R,0)^T \Delta r, t\right) && \text{(12a)}\\
\tilde{\tilde{x}}\left(\mathbf{r} + (1,0)^T \Delta r, t\right) &= \tilde{\tilde{x}}(\mathbf{r}, t) + \tilde{x}\left(\mathbf{r} + (0,R+1)^T \Delta r, t\right) - \tilde{x}\left(\mathbf{r} - (0,R)^T \Delta r, t\right) && \text{(12b)}\\
x\left(\mathbf{r}, t + \Delta t\right) &= a\; \tilde{\tilde{x}}(\mathbf{r}, t), && \text{(12c)}
\end{aligned}$$

which simply means that we first calculate a moving sum in x-direction, then in y-direction, and finally renormalize. Of course, the recursions need to be set up with proper initializations at the boundaries. With this method we need only four additions per cell instead of $(2R+1)^2$ to calculate the local sum.

The method of moving averages can be generalized to use other directions than the Cartesian directions, e.g., the diagonals. It can also be generalized to three or more dimensions.

In the two-dimensional cellular automata we use the average in the two coordinate axes, which leads to a square region of averaging. For different species, we often use different radii R, and thus obtain different diffusion coefficients (since Δr and Δt are the same for all species).

Reactive Step for Moving Average CA

The second part of the new cellular automaton simulates the reaction. It also incorporates the rescaling of the result of the convolution operation. So far in the description of the automaton we have not mentioned the discretization or scaling of the variables. This is not necessary for the description of the diffusion operation, since diffusion (and the convolution which implements it) are linear operations. To be able to call the simulation method a cellular automaton, and to be able to use a lookup table instead of calculating a nonlinear reaction term, we discretize the variables. This means that each variable $x_s(\mathbf{r}, t)$ is an integer in the range $[0, M_s]$. The number of discretization levels can be different for different variables x_s, and usually it is between $M_s = 1$ and $M_s = 100$. Using this discretization it is important to verify that the CA steps respect the discretization, i.e., that the outcome of an operation on the integer variables is again an integer. This is the case for the convolution with a mask where all the entries are integers, e.g., $a_{i,j} = 1$. The non-normalized result is also an integer, now in the range $[0, M_s c_0]$. The normalization, i.e., the multiplication by $1/c_0$ necessary to limit the range to $[0, M_s]$, does not preserve the cell values as integers. That is why it is integrated into the reactive step, which changes the variables by arbitrary non-integer amounts $\Delta t\, f(x)$ anyway. At this step some mechanism has to be introduced to ensure that the result of the reaction step is an integer in the permitted range.

To clarify the separation into two steps, we first discretize the reaction-diffusion equation to be simulated,

$$\frac{\partial x(\mathbf{r},t)}{\partial t} = D\nabla^2 x(\mathbf{r},t) + f(x(\mathbf{r},t)) \tag{13}$$

as

$$x(\mathbf{r}, t+\Delta t) = x(\mathbf{r},t) + \Delta t D \nabla^2 x(\mathbf{r},t) + \Delta t f(x(\mathbf{r},t)) + \mathrm{O}(\Delta t^2). \tag{14}$$

The result of the convolution operation applied to $x(\mathbf{r},t)$ is approximately

$$\Phi_D(x(\mathbf{r},t)) = c_0 x(\mathbf{r},t) + c_2 \nabla^2 x(\mathbf{r},t). \tag{15}$$

We define the operator for the reactive part as

$$\Phi_R(c_0 x(\mathbf{r},t)) = x(\mathbf{r},t) + \Delta t \tilde{f}(x(\mathbf{r},t)). \tag{16}$$

The sequential application of Φ_D and Φ_R results in

$$\begin{aligned}
&\Phi_R(\Phi_D(x(\mathbf{r},t))) \\
&\quad = \Phi_R\left(c_0 x(\mathbf{r},t) + c_2 \nabla^2 x(\mathbf{r},t)\right) && \text{(17a)}\\
&\quad = x(\mathbf{r},t) + \frac{c_2}{c_0}\nabla^2 x(\mathbf{r},t) + \Delta t \tilde{f}\left(x(\mathbf{r},t) + \frac{c_2}{c_0}\nabla^2 x(\mathbf{r},t)\right) && \text{(17b)}\\
&\quad = x(\mathbf{r},t) + \frac{c_2}{c_0}\nabla^2 x(\mathbf{r},t) + \Delta t \tilde{f}\left(x(\mathbf{r},t)\right) + \mathrm{O}(\Delta t^2), && \text{(17c)}
\end{aligned}$$

which we can identify with Eq. (14) if $\tilde{f} = f$ and $\frac{c_2}{c_0} = \Delta t D$. The operator Φ_D in Eq. (15) acts on an array of integers in the range $[0, M_s]$ to give an array of integers in the range $[0, M_s c_0]$. The normalization is contained in the operator Φ_R. As defined, the output of operator Φ_R is not an integer number. We introduce another operator Φ_T for truncation, which takes the real numbers which result from operator Φ_R and produces an integer. The complete dynamics of the CA is given by $\Phi_T \circ \Phi_R \circ \Phi_D$.

In the following we consider several different possibilities for the truncation operator Φ_T.

Discretization operators

There are two ways to present the effect of the discretized reaction step $\Phi_T \circ \Phi_R$. The first is to plot $\Phi_T(\Phi_R(c_0\, x))$ (or the average $\langle \Phi_T(\Phi_R(c_0\, x))\rangle$ for probabilistic Φ_T) directly. The other possibility is to fill a lattice with a spatially homogeneous distribution of states with average value x, and then apply $\Phi_T \circ \Phi_R \circ \Phi_D$, i.e., one iteration of the cellular automaton, to the whole lattice and plot the resulting concentration $h(x)$ as a function of the input concentration x. These two results differ by a small amount due to the fluctuations.

If we fill a lattice with average concentration x and the smallest possible variance, then each node has value $\lfloor x \rfloor$ with probability $1-p$ and $\lfloor x \rfloor + 1$ with probability p, where $p = x - \lfloor x \rfloor$

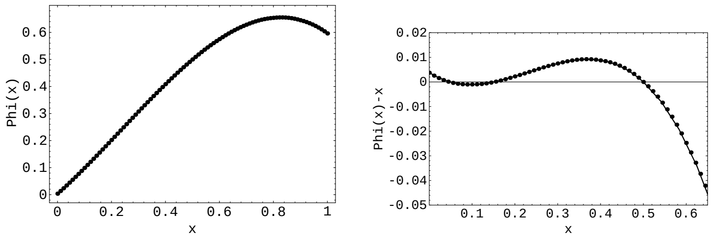

Figure 1: Function $\Phi_R(x)$ (left) and $\Phi_R(x) - x$ (right) for the example system $f(x) = -(x - 0.05)(x - 0.15)(x - 0.5)$ with $M = 10$ and $c_0 = 9$.

(Here and throughout, $\lfloor x \rfloor$ is the largest integer not larger than x). This means that the output of operator Φ_D, i.e., the sum of c_0 local neighbors, has value $c_0 \lfloor x \rfloor + i$ with probability $\binom{c_0}{i} p^i (1-p)^{c_0 - i}, \quad i = 0, \cdots, c_0$. Applying $\Phi_T \circ \Phi_R \circ \Phi_D$ to this lattice and taking the average over the whole lattice, we get

$$h(x) = \sum_{i=0}^{c_0} \binom{c_0}{i} p^i (1-p)^{c_0 - i} \langle \Phi_T(\Phi_R(c_0 \lfloor x \rfloor + i)) \rangle \,, \tag{18}$$

which in general is different from $\langle \Phi_T(\Phi_R(c_0\, x)) \rangle$. To make $h(x) = x + \Delta t f(x)$ (as required by the first order finite difference approximation to the differential equation), we have to construct Φ_R in a special way (which is described in more detail in [9]). We set

$$\Phi_R(c_0\, x) = x + \Delta t f(x) + \Delta t \, \frac{p(1-p)}{2c_0} \, \frac{d^2 f(x)}{dx^2}. \tag{19}$$

The correction term is proportional to $1/M_s$, as opposed to the $f(x)$ term, which is proportional to M_s. Thus the correction term can be neglected for large M_s. Instead of approximating $x + \Delta t f(x)$, we could also approximate

$$h(x) = x + \int_0^{\Delta t} f(x(\tau)) d\tau \tag{20}$$

in order to obtain an approximation to higher order in Δt for the reactive term. But note that the diffusive term remains approximated to $O(\Delta t)$ and $O(\Delta x^2)$. Figure 1 shows the functions involved for an example system which we use to demonstrate the different discretization operators.

We now describe the different discretization methods in detail.

Probabilistic Minimal Noise Rule

The probabilistic minimal noise method is the most useful rule for simulations when macroscopic phenomena are of interest. We define the probability $p = x - \lfloor x \rfloor$ to be used in this and other probabilistic rules. The truncation operator is defined by

$$\Phi_T(x) = \begin{cases} \lfloor x \rfloor & \text{with probability} \quad 1 - p \\ \lfloor x \rfloor + 1 & \text{with probability} \quad p. \end{cases} \tag{21}$$

If the CA rule is implemented using lookup tables for the operation $\Phi_T \circ \Phi_R$, two tables are necessary for this rule, one giving $\lfloor x \rfloor$, and one giving p, for each possible output of the moving average operator Φ_D.

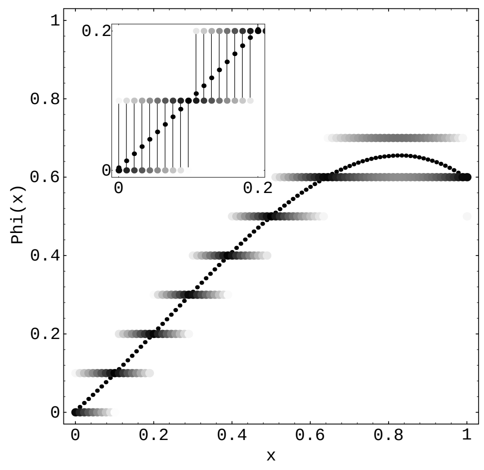

Figure 2: Graphical representation of the probabilistic rule for the example in Figure 1. Large dots mark the possible outcomes, with the grey-level indicating the probability. Small dots mark the average.

Probabilistic Rule for M=1

In the case where $M = 1$, i.e., only the values 0 and 1 are permitted for a cell, $p = x$, and the truncation is simplified to

$$\Phi_T(x) = \begin{cases} 0 & \text{with probability} \quad 1-x \\ 1 & \text{with probability} \quad x. \end{cases} \tag{22}$$

Only the value of x is stored in a table. In this case it is especially important that the correction in Eq. (19) is applied, since it can only be neglected for large M.

Probabilistic Rule with controlled Noise

It is often desirable to have a control over the noise generated by the reactive step. The noise generated by the probabilistic minimal noise rule varies strongly with x. To control the noise level, we set

$$\Phi_T(x) = \begin{cases} \lfloor x \rfloor - n & \text{with prob.} \quad (M_s^2 v_s - m\,p) \,/\, (n(n+m)); \\ \lfloor x \rfloor + m & \text{with prob.} \quad (M_s^2 v_s + n\,p) \,/\, (m(n+m)); \\ \lfloor x \rfloor & \text{otherwise.} \end{cases} \tag{23}$$

where v_s controls the variance and n and m are integers suitably selected so that all probabilities are non-negative.

The resulting average is x, same as for the probabilistic rule with minimal noise. The variance is controlled by $M_s^2 v_s$, which can also be functions of x. At the limits $x = 0$ and $x = M$, the noise must necessarily be zero.

In this case the tables need to store three possible outcomes and two probabilities for each input. For more than one species ($s > 1$) this method also generalizes, and we need to store at least $2s + 1$ outcomes and $2s$ probabilities. A simple selection method based on a binary search is used to keep the method efficient (see [9]).

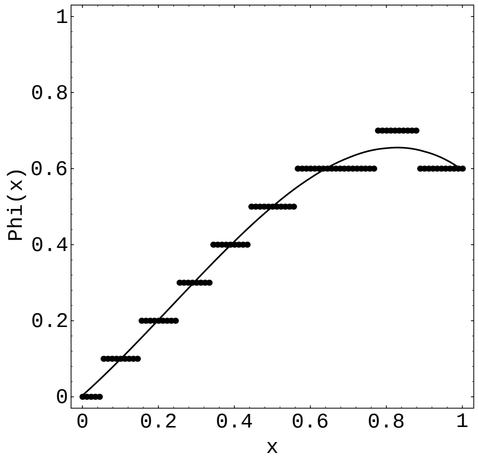

Figure 3: Deterministic rule with simple roundoff for the example. Dots mark the outcomes (deterministic!), and the curve marks $x + \Delta t f(x)$.

Deterministic methods

Since generating one random number for each cell in each time step is a computationally expensive operation, it would be advantageous to have a deterministic truncation. The most obvious choice would be

$$\Phi_T(x) = \lfloor x + 0.5 \rfloor, \tag{24}$$

i.e., ordinary rounding to the next nearest integer. However, this method does not work well in any of the cases we have tried.

Better results can be obtained using a method we call error diffusion. This method derives from the idea that a roundoff error at one cell can be compensated at a neighboring cell, since in the subsequent diffusion step, neighboring cells are again averaged together. To obtain a deterministic rule which does this, we observe that for concentration fields which vary slowly in space, nearby cells will have results of Φ_D which are also close to one another in phase space. Thus we perform the compensation of roundoff errors in phase space: The algorithm to construct the rule table for $\Phi_T \circ \Phi_R$, where `out[a]` is the outcome of $\Phi_T \circ \Phi_R$ given $\mathtt{a} = \Phi_D(x)$ is shown in pseudo-code:

```
err :=0;
for  a:= 0 to M * c0 do
    temp  := PhiR(a) - err;
    out[a]:= Round( temp );
    err   := out[a] - temp;
od
```

The algorithm can easily be generalized to more species, where more possibilities exist to distribute the error in the s-dimensional phase space. Care must be taken that no spurious steady states appear, i.e., values of `a` for which `a = c0*out[a]`, since here whole regions can be locked into one value `a` even if the rate law $f(\mathtt{a})$ is not zero.

In these deterministic models, randomness is introduced in the initial conditions and assumed to persist to some degree during the evolution of the cellular automaton. This is not always the case. Additional sources of randomness can be introduced by integrating another cellular

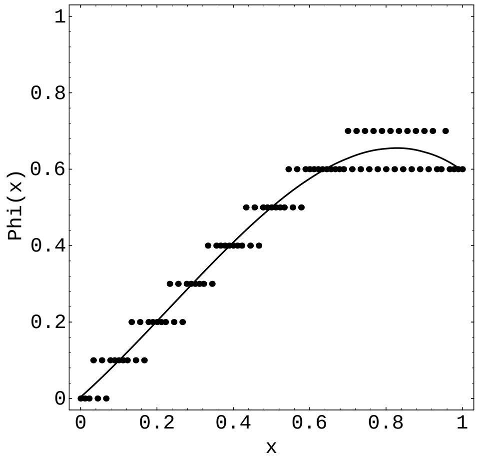

Figure 4: Deterministic rule with error diffusion for the example. Dots mark the outcomes (deterministic!), and the curve marks $x + \Delta t f(x)$.

automaton into the rule. An extra bit could be simulated which uses a CA rule that exhibits random behavior.

Additional noise can be introduced by adding a random contribution to `temp` before rounding off:

$$\texttt{out[a]} := \texttt{Round}(\texttt{temp} + \texttt{noise} * (\texttt{random}() - 0.5)); \tag{25}$$

thus making the difference between neighboring values of `out[a]` bigger than required by the error-diffusion.

Examples of Moving Average CA

We now present a number of examples of simulating reaction-diffusion systems with the moving average CA. The examples are systems that show interesting nonlinear behavior, such as bistability, wavefronts, spatial patterns, etc.

The example applications shown here are used because they are fairly simple, nonlinear, and results can be compared with other CA methods to simulate these or similar systems. The Schlögl model has also been used in the first reactive lattice gas automata [2]. Those automata concentrate on the fluctuations and are very inefficient for macroscopic simulations. The second system is a typical example for excitable media, which have been modeled in many instances in a qualitative way (starting with [6]). In contrast to these automata, which can capture the wave propagation phenomena qualitatively, the systematic construction of CA from the reaction-diffusion system presented here has the advantage that it is applicable to all R-D equations, and that the solutions are quantitatively correct.

Schlögl Model

First we consider a one-species model, namely the Schlögl model [8] in the bistable regime, which is described macroscopically by the cubic rate law (the nonlinear term in the reaction-diffusion equation (13))

$$f(x) = -k(x - x_0)(x - x_1)(x - x_2). \tag{26}$$

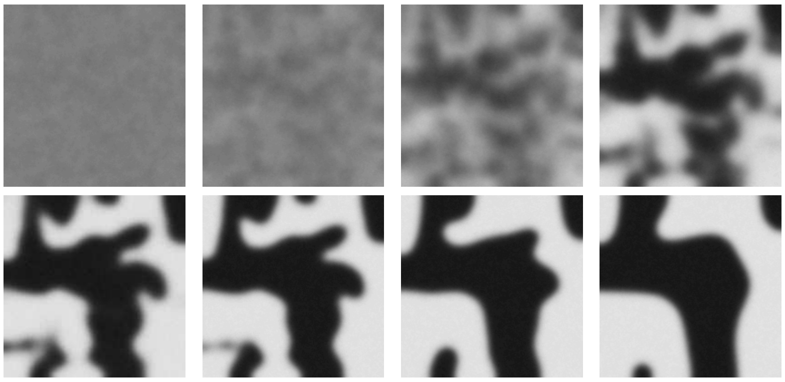

Figure 5: Schlögl model simulated using the moving-average CA with probabilistic rules. Snapshots at $t = 50, 100, 150, 200, 250, 300, 500, 1000$, $\Delta t = 0.2$, size 200×200 cells, 55 discretization levels.

We simulate the symmetric case $(x_1 - x_0) = (x_2 - x_1)$, where we use $x_0 = 0.1$, $x_1 = 0.5$, $x_2 = 0.9$. The parameter k, together with Δt and the size of the neighborhood we use (3×3) determine the space and time scales. Using a probabilistic rule and starting with a uniform initial condition $x = 0.5$, we can observe a departure from the unstable steady state $x_1 = 0.5$. In different regions of space concentrations develop to different values of x, and a separation of regions with $x = x_0$ and $x = x_2$ takes place. Once regions with the two stable states have developed, the dynamics is determined by the movement of the interface between the regions, which can be approximated by

$$N = c + DK \tag{27}$$

where N is the velocity of the interface in the normal direction, c is the speed of a planar interface (here $c = 0$), D the diffusion coefficient ($D = 1/3$ in lattice time and space units), and K the curvature of the interface. This linear dependence can be derived analytically [12] and verified in the simulations [9]. A sequence of images is shown in Figure 5.

Note that if we use the probabilistic minimal noise rule and an odd number of discretized states, e.g., $0 \cdots 50$, then the unstable steady state $x_1 \cdot 50 = 25$ remains unchanged for uniform initial conditions, since in this case there is no noise to force a departure from the steady state. If we have an even number of discretization levels (e.g., $0 \cdots 51$), then $x_1 \cdot 51 = 25.5$ can not be represented exactly, and automatically we have a small deviation from the steady state, which leads to the development of the two stable steady states.

Similar, but worse problems appear with the deterministic rules: In this case spurious steady states appear near both the stable and the unstable steady states. As an example, we compare in Figure 6 the different rules with the following initial conditions: $x(r, 0) = \frac{r}{L}$, i.e., a gradient in concentration. We show the resulting horizontal concentration profile after 100 time steps. The result of the probabilistic simulation corresponds to the exact solution (apart from the small fluctuations). The discretization problems depend also on Δt: for small Δt, the difference between $x + \Delta t f(x)$ and x is so small that after truncation by Φ_T, the result remains the same, and in this way a spurious steady state (where $f(x)$ seems to be zero) appears. To avoid these problems, the probabilistic rule should be used when reliability or precision is required. The deterministic method is faster and could be used in combination with the probabilistic method to speed up transients.

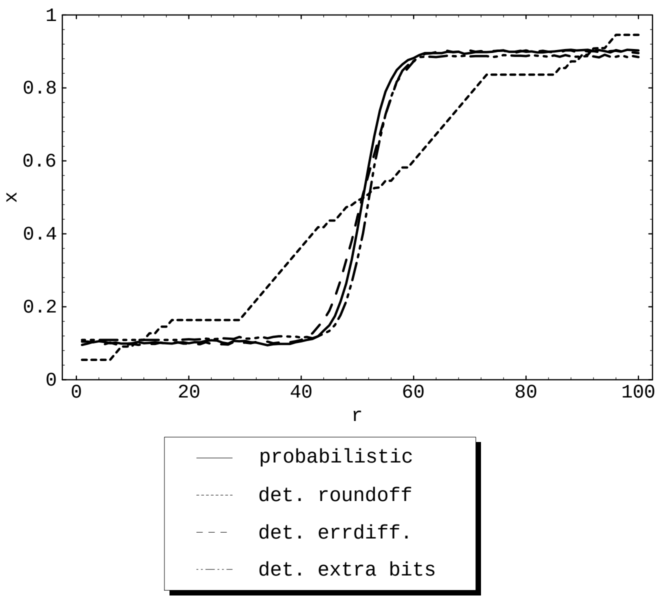

Figure 6: Schlögl model with different rounding methods.

FitzHugh-Nagumo Model – Excitable Media

As an example for excitable media we use the FitzHugh-Nagumo model [4, 7]. This is a two-variable reaction-diffusion system derived from the Hodgkin-Huxley model for nerve-impulse propagation. Of the two variables, one (u) corresponds to the electric potential across the membrane of the nerve cell. This variable changes rapidly and has a large diffusion coefficient. The second variable (v) corresponds to ion concentrations, which change slowly and have a small diffusion coefficient (which we set to zero in some cases).

There are many different ways to write and re-scale the equations for the FitzHugh-Nagumo model.

The essential feature of this model is that the evolution of u is a cubic function of u and contains an additive linear term of v. The evolution of v has one term proportional to u and one term proportional to v. The equations we use here are

$$\dot{u} = D_u \nabla^2 u + (a-u)(u-1)u - v \tag{28a}$$

$$\dot{v} = D_v \nabla^2 v + e(bu - v) \tag{28b}$$

with $a < 1$, $e > 0$, and $b \geq 0$.

The reactive dynamics contains three free parameters, related to the position of the intersection of the u- and v-nullcline (here a), the slope of the v-nullcline (here b), and the relative speed of the evolution of u and v (here e) (The nullclines are the curves in phase space where the reactive parts of $\dot{u}$ and $\dot{v}$ in Eq. (28) are zero).

The homogeneous solutions are oscillatory for $a < 0$. For $a \geq 0$, there can be one or two steady states. We only consider $a > 0$. In this case the state $u_1 = 0, v_1 = 0$ is always stable. For $b < (1-a)^2/4$ there exists another stable state at $u_3 = \left(1 + a + \sqrt{(1-a)^2 - 4b}\right)/2$ and $v_3 = b\,u_3$, as well as an unstable steady state at $u_2 = \left(1 + a - \sqrt{(1-a)^2 - 4b}\right)/2$ and $v_2 = b\,u_2$.

The characteristic feature of excitable media is that if the system is perturbed away from the *stable* steady state more than a small threshold, the return to the unique stable steady state does not follow a direct path, but proceeds via a long excursion in phase space. In the presence

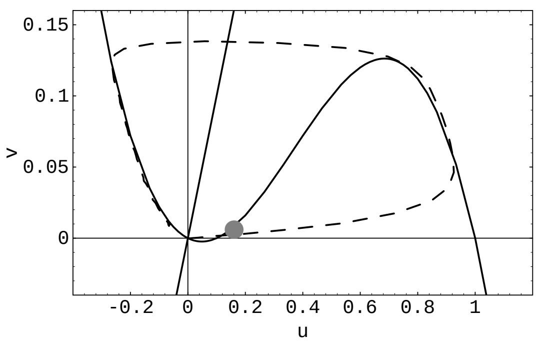

Figure 7: Phase portrait of the FitzHugh-Nagumo model. The nullclines are the solid lines, and the dashed line is the trajectory of one point as the wave passes. The grey point indicates a special initial condition used for Figure 8.

of diffusion, this long excursion allows adjoining regions to cross the excitability threshold which gives rise to traveling waves. Since there is only one stable steady state, these waves are not transitions between one state and another (as in the bistable Schlögl model), but a transitory perturbation. This allows waves to traverse the same region repeatedly. Examples are waves that travel perpetually on one-dimensional circular domains, or spiral waves in two or three dimensions. Such a spiral wave can be initiated by disrupting a planar wave, e.g. by setting a region at the wave front back to the steady state. Another possibility to initiate spiral waves is to find initial conditions that will spontaneously develop into spiral waves even if fluctuations are very small. We can expect that an initial condition that gives rise to spontaneous spirals would lie on the unstable branch of the nullcline for x, so that some points will move to the positive (excited) branch, and other points will move directly to the negative (recovering) branch of the u-nullcline. However, there is no way to indicate which point on the unstable nullcline should be chosen. We can find such a point using methods similar to a bisection method for finding roots of a function [9]. For the parameters used here, the point $u = 0.16$, $v = 0.006$ was found to be a good candidate for this initial condition. If we start with uniform initial conditions of this value, many pairs of spirals develop due to the small fluctuations introduced by the probabilistic rounding. For other initial conditions the fluctuations are much too small to have any macroscopic effect. A large example with many spirals is shown in Figure 8.

If we increase the fluctuations introduced by the roundoff operation, we can arrive at a situation where a nucleation on a circular wave can appear spontaneously even without special initial conditions. An example is shown in Figure 9.

Parallel Implementation with Load-balancing

The cellular automaton has been implemented for simulation on a network of workstations using the message-passing system PVM for communication. The program is organized using the master-slave concept: The master process interacts with the user via a comand interpreter and some visualization processes, and sends data and instructions to one or more slave processes. The slaves perform the actual CA updating, and exchange data on the boundaries directly with each other. For parallel processing we use a one-dimensional decomposition of the two-dimensional domain. This is more efficient than a two-dimensional decomposition, since in the 2-d case a scatter/gather operation has to be performed in order to compose the messages that exchange information at the boundaries, and twice the number of messages are sent, which reduces the

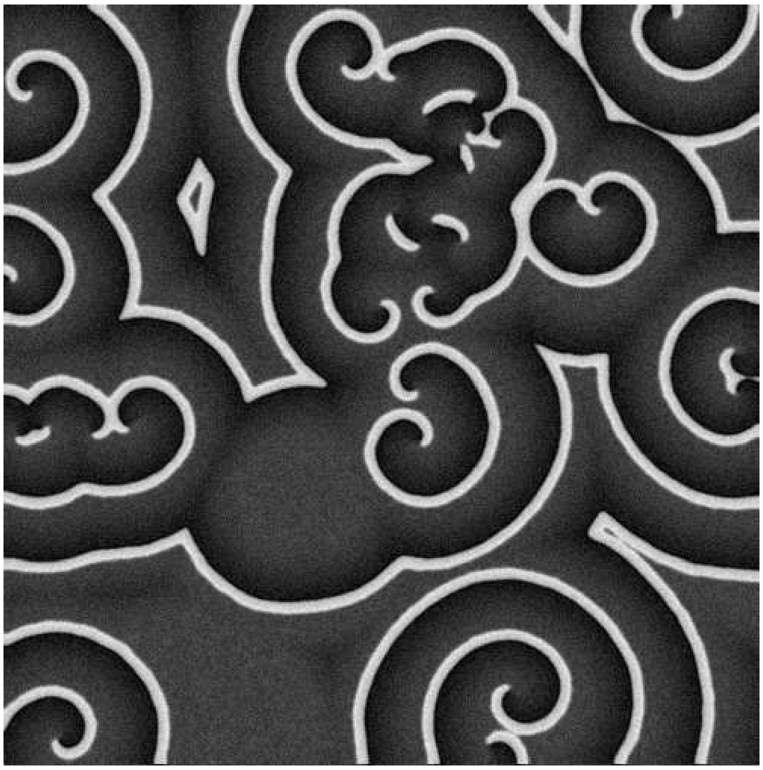

Figure 8: Many spirals developed from the initial conditions $u = 0.16, v = 0.006$ in a large (500×500) system with $\Delta t = 2$, after 500 time steps (showing variable u). The CA uses the probabilistic roundoff method with 55 states for u and 256 states for v.

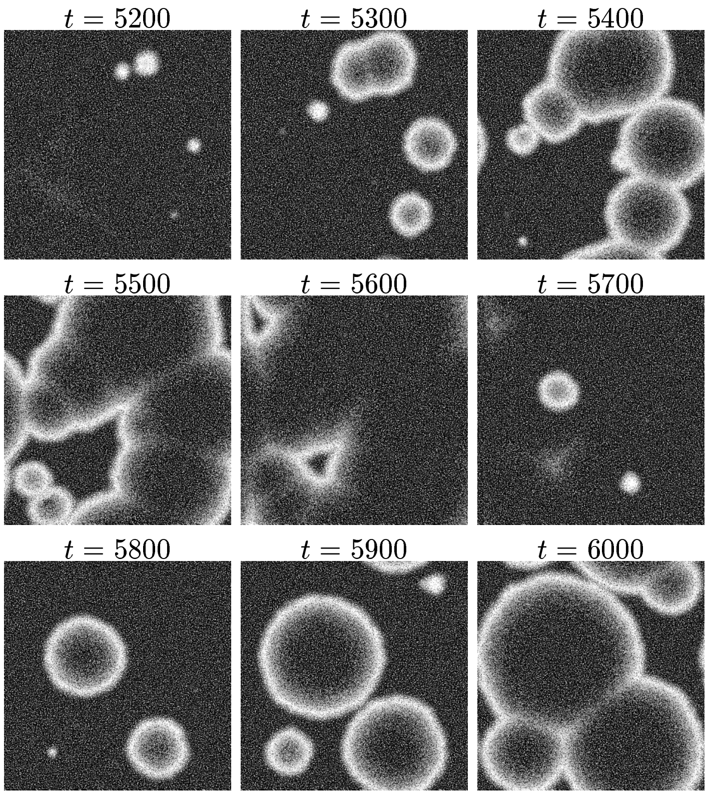

Figure 9: Repeated spontaneous nucleation in the FitzHugh-Nagumo model with noise. The system size is 300×300 cells, $\Delta t = 1$, and the noise level is $s^2 = 0.0013$ for both variables (variable v shown).

performance in this case since message startup times are rather long.

In a heterogeneous network of (nondedicated) workstations, it is important to distribute the load properly. We have implemented a dynamic load balancing which works as follows: The master instructs the slaves to perform a certain number (N) of CA steps. For each step, the slaves calculate the local update function and exchange data on the boundaries. To determine the relative speed and load, each slave measures the time taken for the data interchange step, from the time the first message (of two) is prepared for sending, till both messages are received from the neighbors. This time consists of message startup times, but mostly of time spent waiting for messages from the neighbors. After N update steps (usually $N = 100$), the master collects the data, receiving one slice from each slave, and also collects the waiting times for each slave. In addition the master has measured the total time taken for the N steps.

To calculate the new load we determine the relative speed of each slave. Let w_i be the waiting and communication time (per iteration) of slave i, n_i the width of the slice that slave i works on, and t the average length of one iteration as seen by the master. We then calculate the relative calculation speed of each slave as

$$s'_i = \frac{n_i}{t - w_i} \tag{29}$$

which is then normalized to a sum of one by calculating $s' = \sum_i s'_i$ and $s_i = s'_i/s'$. For the next set of updates, we distribute the work according to

$$n_i = n\ s_i. \tag{30}$$

This distribution of work minimizes the time slave processes spend waiting for data from the (possibly much slower) neighbors. Since load balancing takes place regularly, this scheme adapts to changing machine loads automatically. Of course, due to the centralized load calculation, it does not scale up to arbitrarily large numbers of processors.

Unfortunately, the balance between computation and communication is such that on workstations connected by ethernet, a speedup can only be achieved for very large simulations (e.g., more than 500×500 cells). In most cases we found speedups of less than one. Nevertheless, the master-slave concept with only one slave is very efficient and allows considerable flexibility in utilizing remote workstations while keeping the master process, and thus the I/O and visualization process, on the local workstation.

Conclusion

We have presented a systematic construction of cellular automata for the simulation of reaction-diffusion systems. These cellular automata have been shown to produce results that are in quantitative agreement with analytical solutions and standard numerical techniques. The cellular automata use a moving average technique to implement the diffusive dynamics, and a probabilistic rounding to implement the reactive part. The probabilistic rounding leads to results that are exact *on average*, thereby allowing the strong discretization that is implied by the definition of cellular automata. The simulation technique is especially efficient for complicated reactive terms, since the reactive step is implemented as a table lookup, and the complete nonlinear function is only evaluated to initialize the table. As a massively parallel model, the cellular automata simulation can be easily parallelized, and adaptive load-balancing has been implemented for the simulation on workstations.

Acknowledgements

Part of this work was performed while I was at the Université Libre de Bruxelles in the group of J.-P. Boon. I would like to thank the anonymous referee and T. Worsch for detailed comments on the first draft.

References

[1] Jean Pierre Boon, David Dab, Raymond Kapral, and Anna Lawniczak. Lattice gas automata for reactive systems. Physics Reports, 1996. (to appear).

[2] David Dab, Anna Lawniczak, Jean-Pierre Boon, and Raymond Kapral. Cellular-automaton model for reactive systems. *Phys. Rev. Lett.*, 64(20):2462–2465, 1990.

[3] G. Bard Ermentrout and Leah Edelstein-Keshet. Cellular automata approaches to biological modeling. *J. Theor. Biology*, 160:97–133, 1993.

[4] R. FitzHugh. Impulse and physiological states in models of nerve membrane. *Biophysics J.*, 1:445–466, 1961.

[5] Martin Gerhardt, Heike Schuster, and John J. Tyson. A cellular automaton model of excitable media including curvature and dispersion. *Science*, 247:1563–1566, 1990.

[6] J. M. Greenberg, B. D. Hassard, and S. P. Hastings. Pattern formation and periodic structures in systems modeled by reaction-diffusion equations. *Bull. Am. Math. Soc.*, 84:1296–1327, 1978.

[7] J. S. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proc. IRE*, 50:2061–2071, 1962.

[8] F. Schlögl. Chemical reaction models for non-equilibrium phase transitions. *Z. Physik*, 253:147–161, 1972.

[9] Jörg R. Weimar. *Cellular Automata for Reactive Systems*. PhD thesis, Université Libre de Bruxelles, Belgium, 1995.

[10] Jörg R. Weimar and Jean-Pierre Boon. Class of cellular automata for reaction-diffusion systems. *Physical Review E*, 49(2):1749–1752, 1994.

[11] Jörg R. Weimar, John J. Tyson, and Layne T. Watson. Third generation cellular automaton for modeling excitable media. *Physica D*, 55:328–339, 1992.

[12] V. S. Zykov. Analytical evaluation of the dependence of the speed of an excitation wave in a two-dimensional excitable medium on the curvature of its front. *Biophysics*, 25:906–911, 1980.