Definition of Cell-DEVS Models for Complex Diffusion Systems

Wei Ding, Xiuping Wu, Laurentiu Checiu, Changshuang Lin, Gabriel Wainer

Department of Systems and Computer Engineering. Carleton University
1125 Colonel By Dr. Ottawa, ON. K1S 5B6. Canada.
gwainer@sce.carleton.ca

Abstract. Cell-DEVS is an extension to the DEVS formalism that allows the definition of cellular models. CD++ is a modeling and simulation tool that implements DEVS and Cell-DEVS. Here, we show the use of these techniques through different application examples. Complex applications can be implemented in a simple fashion, and they can be executed effectively. We introduce different models of physical systems for diffusion applications, including a model of Diffusion Limited Aggregation, Snowflake growth, 3D Reaction-Diffusion and Driven Diffusion.

1. INTRODUCTION

In recent years, many simulation models of real systems have been represented as cell spaces [1, 2]. Cellular Automata [3] is a well-known formalism to describe these systems, defined as infinite n-dimensional lattices of cells whose values are updated according to a local rule. Cell-DEVS [4] was defined as a combination of cellular automata and DEVS (Discrete Event Systems specifications) [5], with the goal of improving execution speed building discrete-event cell spaces, and to improve their definition by making the timing specification more expressive.

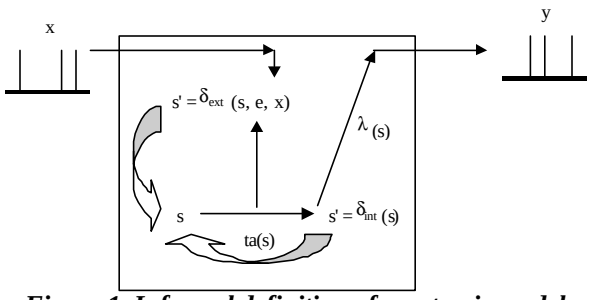

Figure 1. Informal definition of an atomic model

DEVS [5] provides an abstract approach of creating hierarchical and modular models, separating the modeling from the simulation. The basic building block of any DEVS model is the *atomic* model, which can be connected to others to form what is called a *coupled* model. A DEVS atomic model can be informally described as in Figure 1.

Each atomic model can receive *inputs* (x) and generates *outputs* (y) from/to other models. The *state* (**s**) of the model is associated with a *time advance* (**ta**) function, which determines its duration. Once this time is consumed, an internal transition happens. At that moment, the model execution results are spread through the model's output ports by activating an *output function* (l). Then, an *internal transition function* (d_{int}) is fired, producing a local state change. External input events (received from other models) trigger the external transition function (d_{ext}), which specifies how to react to them.

A DEVS coupled model is composed of several atomic or coupled sub-models, as shown in Figure 2.

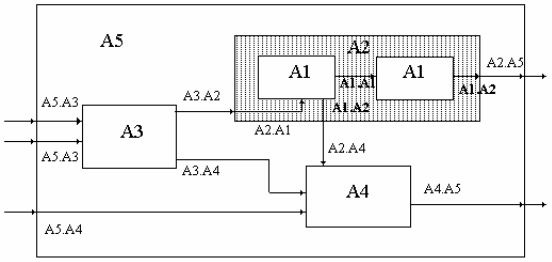

Figure 2. Informal description of a coupled model

Coupled models are defined as a set of basic components (atomic or coupled), which are interconnected through the model interfaces. The model's coupling scheme defines the interconnectivity between models and the interface with the external world.

Cell-DEVS 0 has extended DEVS, allowing the implementation of cellular models with timing delays. Each cell is defined as a DEVS atomic model, and it can be later integrated to a coupled model representing the cell space. Cell-DEVS atomic models can be described as in Figure 3.

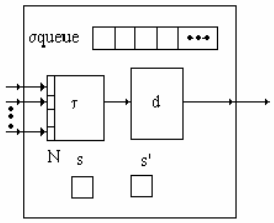

Figure 3. Cell-DEVS atomic model

Each cell uses N inputs (from its neighborhood) to compute its next state. These inputs, which are received through the model's interface, activate a local computing function (t). A delay (**d**) can be associated with each cell. The state (**s**) changes can be transmitted to other models, but only after the consumption of this delay. Two kinds of delays can be defined: *transport* delays model a variable commuting time, and *inertial* delays, which have preemptive semantics (scheduled events can be discarded if the computed value is different than the future state).

Once the cell behavior is defined, a coupled Cell-DEVS can be created by putting together a number of cells interconnected with its neighbors. A sample Cell-DEVS coupled model is presented in Figure 4. A coupled Cell-DEVS is composed of an array of atomic cells, with given size and dimensions. Each cell is connected to its neighborhood through standard DEVS input/output ports. Border cells have different behavior due to their particular locations, which may result in a non-uniform neighborhood.

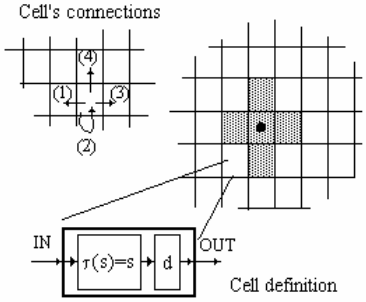

Figure 4. Cell-DEVS coupled model

The CD++ tool 0 was developed following the definitions of the Cell-DEVS formalism. CD++ is a tool to simulate both DEVS and Cell-DEVS models. Cell-DEVS are described using a built-in specification language, which provides a set of primitives to define model's behavior, following the descriptions presented in figures 1-4. A built-in language permits defining the t function of the formal specification using a set of rules. The language has a large collection of functions and operators. The most common operators are included: Boolean, comparison, and arithmetic; trigonometric, roots, power, rounding and truncation, module, logarithm, absolute value, minimum, maximum, G.C.D., L.C.M., etc.

We will show how to apply Cell-DEVS to simulate a variety of diffusion systems. Reaction-diffusion is a process in which two or more chemicals diffuse over a surface and react with one another to produce stable patterns. Reaction-diffusion can produce a variety of spot and stripe patterns, and are some of the most popular cellular models available. We describe different models that are implemented and executed using the CD++ tool, focusing on particular characteristics of each system. We will show how complex applications can be implemented in a simple fashion using the advanced features provided by Cell-DEVS and the CD++ implementation

2. DIFFUSION LIMITED AGGREGATION

Diffusion Limited Aggregation (DLA) occurs when diffusing particles stick to and progressively enlarge an initial seed represented by a fixed object. The seed typically grows in an irregular shape resembling frost on a window 0. Diffusion is a random motion with respect to the direction. There are two kinds of particles in a grid: fixed (seeds) and mobile. A mobile particle has same probability of walking toward each direction. When a mobile particle finds a seed, it sticks to the fixed particle, and it becomes fixed. In other words, they form aggregates. A mobile particle disappears it strays too far from the center. The process continues until all mobile particles either have disappeared or they have become fixed. Some illustrations of DLA can be found in 0.

We built a Cell-DEVS model of DLA, as follows. The DLA model was implemented as a 2D Cell-DEVS. We will briefly present some of the rules we used (details can be found in 0). Initially, a certain percentage of the cells are occupied by mobile particles, and there are at least one or more seeds. The system evolves with the following rules.

- A particle can move in four directions (N/S/E/W)
- A particle becomes fixed an adjacent cell contains fixed particles.
- An empty cell will be occupied if there is at least one mobile particle trying to move in, and there is no seed adjacent to the mobile particle.
- In the case that there are more than one particle that intend to move toward a same empty cell, the moving direction is used as priority.
- A mobile particle that cannot move will select a new direction at random.
- A mobile particle disappears if it strays too far from the center.

A cell with a value of 0 indicates is empty; a value of 1 to 4 denotes a mobile particle and its moving direction. A cell with a value of 5 indicates a seed. When a cell is empty, it checks to see if there are any mobile particles wanting to move to that cell. Such mobile particle can move only if it does not have any adjacent seed:

```
rule : {round(uniform(1,4))} 100 { (0,0)=0 and (((0,-1)=2
and (-1, -1) !=5 and (1,-1) !=5 and (0,-2) !=5 ) or
((-1,0)=3 and (-1,-1)!=5 and (-2,0) !=5 and (-1,1) !=5)or
((0,1)=4 and (1,1) !=5 and (0,2) !=5 and (1,1) !=5 ) or
((1,0)=1 and (1,1) !=5 and (2,0) !=5 and (1,-1) !=5 ))  }
```

The following rules illustrate the resolution of moving conflict for a particle that is attempting to move a cell up. A mobile particle with moving direction 1 (up) can move in an empty cell above if there is no other mobile particle that attempts to move in.

```
% direction=1 (up): change direction when nowhere to move
rule : {round(uniform(1,4))} 100 { (0,0)=1 and (-1,0)!=0}
rule : {round(uniform(1,4))} 100 { (0,0)=1 and (-1,0)=0
and
(((-2,0)=3 and (-2,-1)!=5 and (-3,0)!=5 and(-2,1)!=5 ) or
((-1,-1)=2 and (-1,-2)!=5 and(-2,-1)!=5 and (0,-1)!=5 )or
((-1,1)=4 and (-2,1) !=5 and (-1,2) !=5 and (0,1) !=5 ))}
```

Whenever a mobile particle is in a cell with any fixed particle adjacent to it, it becomes fixed.

```
% particle becomes fixed if adjacent cell contains a seed
rule : 5 100 { (0,0)> 0 and (0,0)<5 and
((-1, 0) = 5 or (0, -1) = 5 or (0, 1) = 5 or (1, 0) = 5 )
}
```

Several scenarios were executed with different number of seeds and percentages of concentration (details can be found in 0). The following figure presents a version with a concentration of 30% (grid size: 71x71)

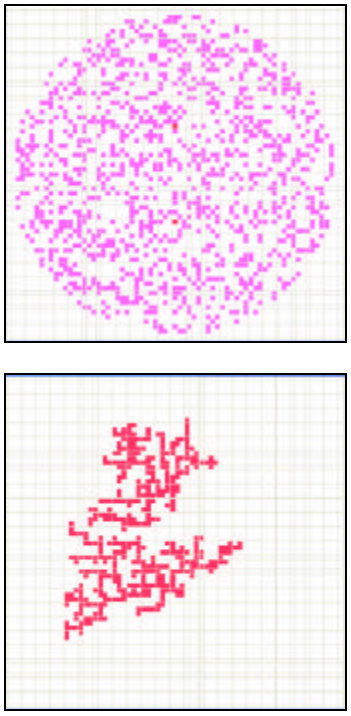

Figure 5: Initial/Final Execution Results (Two seeds and 30% concentration)

The following figure shows similar results for a model with two seeds and a concentration of 40%

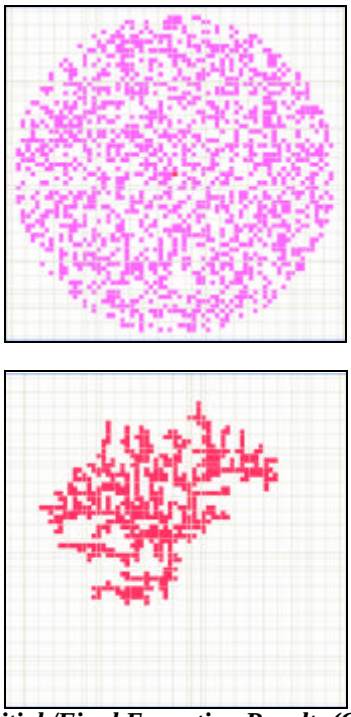

Figure 6: Initial /Final Execution Result: (One seed and 40% concentration)

3. SNOWFLAKE GROWTH SIMULATION

We built a model of Snowflake Growth, based on the Local Cellular Model for Snow Crystal Growth described on 0. The dendrite growth of snowflake is very complex, and influenced by many factors in its natural environment. Here, we will show a simplified 2D Cell-DEVS model with some parameters to influence the desired diversity, and use this model to investigate the impact of varying conditions of growth. The form of snow crystals depends heavily upon the saturation and temperature during growth. The forms observed include dendrites, stellar forms, sectors, plates, needles, spatial, columns, scrolls, etc.

Wolfram popularized a Boolean model for snowflake growth [12], in which, at each time step, each cell is either ice or not. On the subsequent step, cells that were ice, remain ice; instead, cells that were not, become ice if exactly one of the neighboring cells is ice. The model we built is more realistic, but the transition rules we used work mostly on a similar spirit than Wolfram's model.

Our model uses a arrangement of cells containing a real value. We assume the value of any cell as measuring the amount of water at that cellular location in air. Values of one or higher correspond to ice, while lower values are taken to represent water in a form that may possibly move to neighboring cells. Each cell is classified as either receptive

or non-receptive. The first stage is to determine the receptive sites: those that are ice or have an immediate ice neighbor. At the next stage, the values of the cells are given by the values at the receptive sites plus a constant γ plus a diffusion term. The diffusion term is a local average of a modified cellular field obtained by setting the receptive sites to zero. The averaging algorithm as below:

$$Vu = 0.5*Vo + 0.5*?Vn/8 \quad (1)$$

Vu: Update value of Cell
Vo: Original value of cell
Vn: Value of 8 immediate neighbors

Therefore, the center cell has weight 1/2 while the 8 neighboring cells each has a weight of 1/16. The motivation for the model is that receptive sites are viewed as permanently storing any mass that arrives at that point. The mass in the unreceptive sites is free to move, and hence moves toward an average value. Lastly, the constant γ added to receptive sites corresponds to the idea that not only is "add a constant" an especially simple generalization to the model. Informally, this captures the idea that some water may be available from outside the plane of growth. The constant to be added to the receptive sites is one of the parameters that we vary. The second parameter that we vary is the background level β. We begin with a single cell of value one (an ice seed) in a sea of a constant background. We denote the added constant by γ and the background level by β. The boundary conditions are fixed at the background level at a fixed (Euclidean) distance from the initial cell. These boundary conditions attempt to make the boundary conditions as isotropic as possible [3].

All cells can be divided into three types, and we have three rules addressing independently to each cell group.

a. Cells made of ice

```
Rule : {(0,0)+ 0.001} 10 {(0,0)>= 1}
```

Every ice cell will absorb more and more water in the air continuously.

b. Cells that are not ice, and has no ice neighbors
c. Cells that have no ice but have ice neighbors

Following, we present the rules used for the cells that have no ice and no ice neighbors, which are computed based on Equation (1). The rules for Cells with no ice but ice neighbors are equivalent and they are not presented here.

```
rule : {(0,0)*0.5 +(
if((if((-2,2)>=1,1,0)+if((-1,2)>=1,1,0)+
   if((0,2)>=1,1,0)+if((-2,1)>=1,1,0)+if((0,1)>=1,1,0)+
   if((-2,0)>=1,1,0)+if((-1,0)>=1,1,0)) = 0,(-1,1),0)+
 if((if((-1,2)>=1,1,0)+if((0,2)>=1,1,0)+if((1,2)>=1,1,0)+
if((-1,1)>=1,1,0)+if((1,1)>=1,1,0)+if((-1,0)>=1,1,0)+
if((1,0)>=1,1,0))=0,(0,1),0)+if((if((0,2)>=1,1,0)+
if((1,2)>=1,1,0)+if((2,2)>=1,1,0)+if((0,1)>=1,1,0)+
if((2,1)>=1,1,0)+if((1,0)>=1,1,0)+
if((2,0)>=1,1,0))=0,(1,1),0)+
if((if((-2,1)>=1,1,0)+if((-1,1)>=1,1,0)+if((0,1)>=1,1,0)+
if((-2,0)>=1,1,0)+if((-2,-1)>=1,1,0)+if((-1,-1)>=1,1,0)+
if((0,-1)>=1,1,0)) = 0,(-1,0),0)+
if((if((0,1)>=1,1,0)+if((1,1)>=1,1,0)+if((2,1)>=1,1,0)+if
((2,0)>=1,1,0)+if((0,-1)>=1,1,0)+
if((1,-1)>=1,1,0)+if((2,-1)>=1,1,0)) = 0,(1,0),0)+
if((if((-2,0)>=1,1,0)+
if((-1,0)>=1,1,0)+if((-2,-1)>=1,1,0)+
if((0,-1)>=1,1,0)+if((-2,-2)>=1,1,0)+
if((-1,-2)>=1,1,0)+
if((0,-2)>=1,1,0)) = 0,(-1,-1),0)+
if((if((-1,0)>=1,1,0)+if((1,0)>=1,1,0)+
if((-1,-1)>=1,1,0)+if((1,-1)>=1,1,0)+
if((-1,-2)>=1,1,0)+if((0,-2)>=1,1,0)+
if((1,-2)>=1,1,0)) = 0,(0,-1),0)+if((if((1,0)>=1,1,0)+
if((2,0)>=1,1,0)+if((0,-1)>=1,1,0)+if((2,-1)>=1,1,0)+
if((0,-2)>=1,1,0)+if((1,-2)>=1,1,0)+if((2,-2)>=1,1,0))
                = 0,(1,-1),0)  )/16}

10

{(0,0) < 1 and (if((-1,1)>=1,1,0)+if((0,1)>=1,1,0) +
if((1,1)>=1,1,0)+if((-1,0)>=1,1,0)+if((1 ,0)>=1,1,0) +
if((-1,-1)>=1,1,0)+if((0,-1)>=1,1,0)+if((1,-1)>=1,1,0))=0
}
```

The rule computes equation (1) described before. In the rule, *(0,0)*0.5* means the original value of center cell has weight 1/2 of the final updated value of cell. The the another 1/2 weight be contributed by the 8 groups of "nested if" structures, each of them represents the value of one immediate neighbor[(-1,1),(0,1),(1,1),(-1,0),(1,0),(-1,-1),(0,-1),(1,-1)] of the center cell. In each "nested if" structure, the rule will also estimate the values of "neighbor's neighbor" to determine the contribution value of this structure will be zero or unchanged. For instance, if the values of immediate neighbors of cell (-1,1) are all zero, which means they are all non-ice cells, we will let the contribution value of (-1,1) to center cell as unchanged, otherwise, this value will be zero. The rest may be deduced by analogy based on the same principle. After the predefined elapse time, the value of the center cell will be updated by the new value which we computed using this rule.

Based on the detailed description of rules list above, we can find when we calculated the updated value of the particular non-ice cell whatever it has ice neighbor or not, we need to know the value of immediate neighbor's neighbors. Therefore, we introduced the extra outer neighbor other than 8 immediate neighbors for each cell. Furthermore, the one special rule defines the boundary conditions of the cell space will be executed at first. We assume the boundary conditions of cell space will be fixed at the whole process of simulation.

In order to investigate different results from the model by using different variables, we set three different group variable vectors (γ ,β) and size of the cell space 30 x 30.

This model was tested using different models for the three vectors, as shown in the following figure:

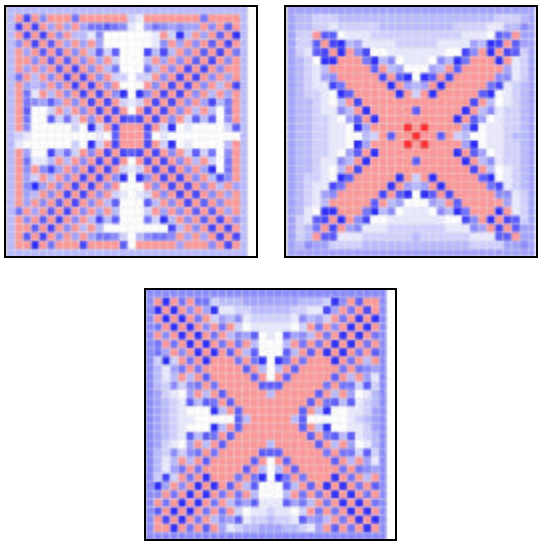

Figure 7. Snowflake formation. 10000 iterations cell space = (30x30) (a) ß = 0.3, ? = 0.001, (b) ß = 0.4, ? = 0.01, (c) ß = 0.05, ? = 0.0035.

4. A 3D MODEL OF REACTION-DIFFUSION

We created a 3D reaction-diffusion model, based on the one presented in 0. Reaction diffusion systems can be described by a set of partial differential equations as follows:

$$x_i' = D_i \nabla^2 x_i + f_i(x_1, x_2, \ldots x_n),\ i=1..n \qquad (2)$$

where the first term represents the diffusion equation and the second term is the reaction equation (f_i are always non-linear; $x_i' = dx_i/dt$). To simulate reaction diffusion systems we apply diffusion first, and then we reaction. In this case, we implemented the models as a 4D diffusion Cell-DEVS.

$$\hat{x} = \frac{1}{cardN} \sum_{xi \in N} xi \quad (3)$$

Then, we apply reaction on this new value. Reaction differential equation is:

$$\dot{x} = f(x) \quad (4)$$

$$\frac{x(t) - x(t - \Delta t)}{\Delta t} = f(x) \quad (4')$$

In this case, we need to know the previous state of the cell so we will use a second 3D hyperplane as memory of the previous state. According to 0 we will use the following structure for neighboring cells.

Figure 8. Neighborhood shape

To do so, we use 4 dimensions in our representation, as we need a 3D hyperplane to memorize the previous state of the system, and the neighborhood shape in figure 8.

```
[rd]
type : cell          dim : (5,5,5,2)
delay : transport    border : nowrapped
neighbors :  (0,0,-1,0) (-1,0,0,0)  (0,0,1,0)
neighbors :  (0,-1,0,0) (0,0,0,0)  (0,1,0,0)
neighbors : (1,0,0,0)  (0,0,0,-1) (0,0,-1,1)
neighbors : (-1,0,0,1) (0,0,1,1)  (0,-1,0,1)
neighbors : (0,0,0,1)  (0,1,0,1) (1,0,0,1)
localtransition : rd-rule

zone : rd-rule { (0,0,0,0)..(4,4,4,0) }
zone : memory-rule { (0,0,0,1)..(4,4,4,1) }

[memory-rule]
rule : {(0,0,0,-1)} 70 { t }

[rd-rule]
rule : {(#macro(diffusion)-(0,0,0,1))/ 100 }
                    100 { t }
```

In the memory rule we save the value from the cell "below" and in the *rd-rule*, we calculate first the diffusion as follows, computed as (4').

```
#BeginMacro(diffusion)
( ((0,0,0,0)+(-1,0,0,0)+(1,0,0,0)+(0,-1,0,0)
   + (0,1,0,0)+ (0,0,1,0)+ (0,0,-1,0) ) / 7 )
#EndMacro
```

Then we substract the previous value of the cell stored in the "above" cell. The *rd-rule* is applied on the reaction diffusion zone { (0,0,0,0)..(4,4,4,0) }, while the *memory-rule*, which saves the previous state of all cells from reaction diffusion zone, is saved in { (0,0,0,1)..(4,4,4,1) }.

The following figures represent the different reaction of two substances. The first group of 5 squares (each square consisted of 5x5 cells) from the first row in Figure 8 represents the first step of our simulation; the next group of 5 squares (5x5 cells) from the same row represents the initial state.

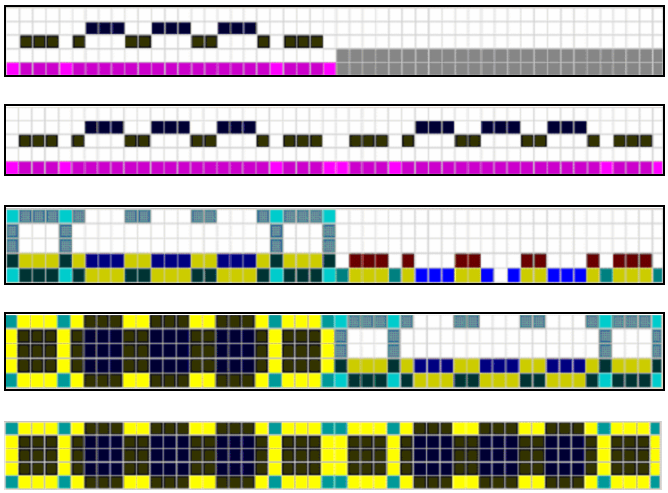

Figure 8. Execution results

The second row in Figure 8 represents the second step in our simulation: first 5 squares are the actual state of the system and the next 5 squares are the previous step. The third row in Figure 8 is the state of the system in the 20th step of simulation. The fourth row is the 34th step of simulation where the system reached the equilibrium, is stable. The fifth row in Figure 8 proves this fact: the current state is the same with the previous state (represented by the memory layer) after indefinite number of simulation steps following the 34th step.

5. DRIVEN DIFFUSION

Our last model describes the random motion of two types of particles in a system under the influence of an external field. The field may drive one species of particles move along the field direction while the other species move against that direction 0. This kind of model can simulate the behavior for certain kinds of materials such as superionic conductors, fast ion conductors, and solid electrolytes. These two species of particles are differentiated by their attribute of positive charge or negative charge. The particle space contains approximately the same amount of positive and negative particles so that the total charge of the system is zero. We created a Cell-DEVS model to simulate the system and to study how the density of the particle space affects the behavior of the system.

Initially, the space is occupied by the two randomly distributed particles A and B and each particle has a randomly chosen direction to face (N/E/S/W). The density of the occupied space is denoted as P. In the case of an external electrical field appearance (assuming the field points to the north-east), the preferable moving direction of particle A is North or East while the preferable moving direction of particle B is South or West. The probability of A and B hops along that preferable direction is *a*, and the probability of hopping against the direction is (1-a).

The rules for a particle of either type to move are as follows:

1) The cells only move towards the direction that they are facing now. The adjacent cell mentioned below refers to the neighboring cell that the particle is facing now.
2) If the adjacent cell is occupied, then the particle remains in its current place and randomly chooses a direction to face.
3) If the adjacent cell is empty but faced by one or more particles, then the particle remains in its current place and randomly chooses a direction to face.
4) If the adjacent cell is empty and faced by no other particles, then the particle moves to the adjacent cell and chooses randomly a direction to face.

Rules for a particle to randomly choose a direction to face are as follows:

- For particle A, the probability of choosing north, east, south or west to face are (a/2), (a/2), (1-a)/2 and (1-a)/2 respectively.
- For particle B, the probability of choosing north, east, south or west to face are (1-a)/2, (1-a)/2, (a/2) and (a/2) respectively.

The rules for updating cells are as follows:

1) If the cell is empty and faced by no particles, then it remains empty.
2) If the cell is empty and faced by exactly one particle, then the cell will be occupied.
3) If the cell is empty and faced by two or more particles, then the cell remains empty.
4) If the cell is occupied and the inside particle faces an empty cell that is not faced by other particles, then this cell will be vacated.

Different tests were carried out, considering different density values, space size, and initial states. Particles are initially distributed at random in the space according to the given density value.

Since the model is used to study how the system behavior are affected by the density of the particles distributed in the space, several test cases have been designed and the results along with their analysis are presented below.

In our first case, about 10% of the cell space is occupied by the randomly distributed (as seen in Fig. 9 a).

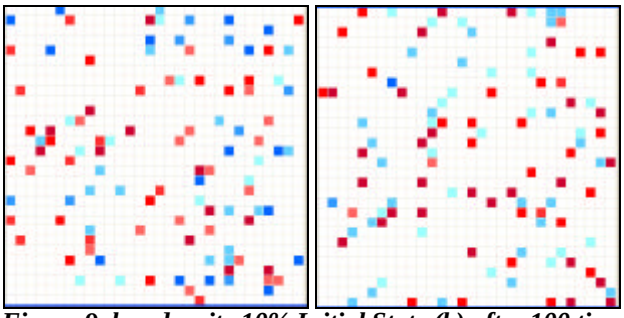

Figure 9. low density 10% Initial State (b) after 100 time steps

After 100 time steps, the particles are still randomly distributed. The cell space remains disordered over the simulated time steps, while the distributions of particles A and B are homogeneous (high current in the system). Similar results were obtained with a density of 20%, as showed in Fig. 10.

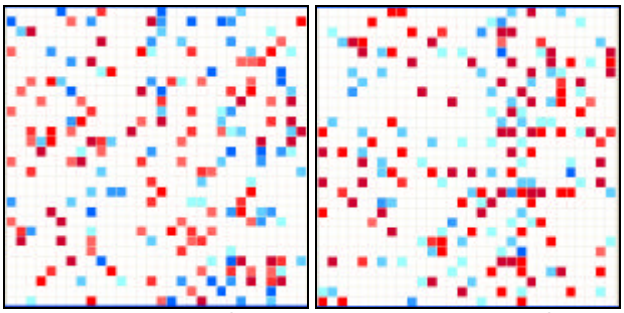

Figure 10. density of 20%. Initial cell space (b) after 100 time steps

Then, we tested a case in which density of the whole space is higher (40%). The initial cell space is illustrated in Figure 11 below.

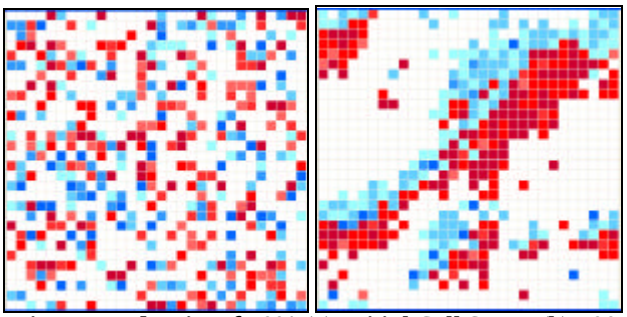

Figure 11. density of 40% (a) Initial Cell Space (b) 100 time steps

From the results at time step 100 as shown in Figure 6, we can see that the distribution of the two particles is now pretty ordered. It exhibits striped, banded structure. Within each strip, there exist two sub-strips with each having approximately the same amount of particles. This indicates the non-homogeneities of the distribution of two particles and thus results in reduced current in the system.

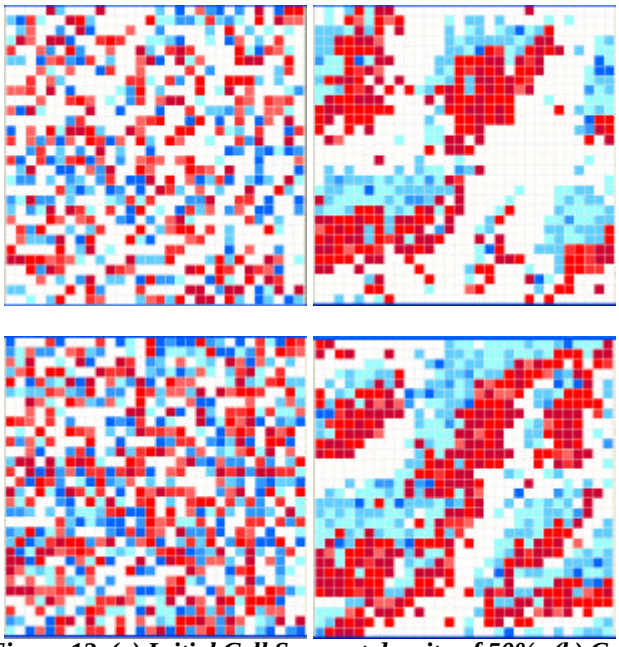

Figure 12. (a) Initial Cell Space at density of 50%; (b) Cell Space at density of 50% after 100 time steps; (c) Initial Cell Space with density of 70%; (d) Cell Space with density of 70% after 100 time steps

6. CONCLUSIONS

Cell–DEVS allows describing complex systems using an n-dimensional cell-based formalism. Complex timing behavior for the cells in the space can be defined using very simple constructions. The CD++ tool, based on the formalism entitles the definition of complex cell-shaped models. We have used CD++ to build a Cell-DEVS to build different diffusion models. Diffusion-limited aggregation is an example of fractal growth model. A snowflake formation model showed how to define pattern formation models. Our Driven Difussion model showed how particles align in a system where external excitation occurs. Finally, our 3D reaction-difussion model showed how to create higher dimensional models in a simple fashion.

The tool and the examples are the public domain and they can be obtained in:

`http://www.sce.carleton.ca/faculty/wainer/`

REFERENCES

[1] D. Talia. "Cellular processing tools for high-performance simulation". IEEE Computer. September 2000. Pp. 44 –52.

[2] M. Sipper. "The emergence of cellular computing". IEEE Computer. July 1999. Pp. 18-26.

[3] Wolfram S. “A new kind of science”. Champaign: Wolfram Media; 2002.

[4] G. Wainer, N. Giambiasi "N-Dimensional Cell-DEVS". In *Discrete Events Systems: Theory and Applications*, Kluwer. Vol. 12, No. 1. January 2002. pp. 135-157.

[5] B. Zeigler, T. Kim, and H. Praehofer. *Theory of Modeling and Simulation*. Academic Press. 2000.

[6] Taylor, M. 1996. *Partial Differential Equations: Basic Theory*. Springer Verlag, NY

[7] G. Wainer. “CD++: a toolkit to develop DEVS models”. *Software-Practice and Exp*. 32, 1261-1306. 2002.

[8] T. Toffoli and N. Margolus, *Cellular Automata Machines: a New Environment for Modeling*, MIT Press, London, England, 1978.

[9] “Five Cellular Automata: Diffusion-Limited Aggregation”. Online document, available on: http://www.hermetic.ch/pca/da.htm

[10] C. Lin; G. Wainer. “Modeling DLA”. Internal Report. Carleton University, Dept. of Systems and Computer Engineering. 2003.

[11] C. Reiter. A Local Cellular Model for Snow Crystal Growth, *Chaos, Solitons & Fractals* 23. 1111-1119 (2005).

[12] Packard NH. Lattice models for solidification and aggregation. Theory and Applications of Cellular Automata. Wolfram S, ed., Singapore: World Scientific Publishing; 1986. p. 305-310.

[13] J. Weimar “Three-dimensional Cellular Automata for Reaction Diffusion Systems”. Fundamenta Informaticae, 52(1–3) pp. 275–282. (2002)

[14] R. J. Gaylord, K. Nishidate, “Modeling Nature”, Springer-Verlag New York, Inc., 1996