Using Cell-DEVS for modeling complex cell spaces

Javier Ameghino[1], Gabriel Wainer[2]

[1]Computer Science Department. Universidad de Buenos Aires
Ciudad Universitaria (1428). Buenos Aires. Argentina.
[2]Department of Systems and Computer Engineering. Carleton University
1125 Colonel By Dr. Ottawa, ON. K1S 5B6. Canada.

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 formalisms. 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 present example models of wave propagation, a predator following prey while avoiding natural obstacles, an evacuation process, and a flock of birds.

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 Events Systems specifications) [5]. The goal is to improve execution speed building discrete-event cell spaces, and to improve their definition by making the timing specification more expressive.

DEVS is a formalism proposed to model discrete events systems, in which a model is built as a composite of basic (behavioral) models called **atomic** that are combined to form **coupled** models. A DEVS atomic model is defined as:

$$M = < X, S, Y, \delta_{INT}, \delta_{EXT}, \lambda, D > \quad (1)$$

where **X** represents a set of input events, **S** a set of states, and **Y** is the output events set. Four functions manage the model behavior: $\boldsymbol{\delta_{INT}}$ the internal transitions, $\boldsymbol{\delta_{EXT}}$ the external transitions, $\boldsymbol{\lambda}$ the outputs, and D the duration of a state. When external events are received, the external transition function is activated. The internal events produce state changes when the lifetime of a state is consumed. At that point, the model can generate outputs, and results are communicated to its influencees using the output ports.

Once an atomic model is defined, it can be incorporated into a coupled model is defined as:

$$CM = < X, Y, D, \{Mi\}, \{Ii\}, \{Zij\} > \quad (2)$$

Each coupled model consists of a set of **D** basic models **Mi**. The list of influences **Ii** of a given model is used to determine the models to which outputs (**Y**) must be sent, and to build the translation function **Zij**, in charge of converting outputs of a model into inputs (**X**) for the others. An index of influences is created for each model (**Ii**). For every j in the index, outputs of model **Mi** are connected to inputs in model **Mj**.

In Cell-DEVS, each cell of a cellular model is defined as an atomic DEVS. Cell-DEVS atomic models are specified as:

$$TDC = < X, Y, S, \theta, N, delay, d, \delta_{INT}, \delta_{EXT}, \tau, \lambda, D > \quad (3)$$

Each cell will use the **N** inputs to compute the future state **S** using the function $\boldsymbol{\tau}$. The new value of the cell is transmitted to the neighbors after the consumption of the delay function. **Delay** defines the kind of delay for the cell, and **d** its duration. The outputs of a cell are transmitted after the consumption of the delay.

Once the cell atomic model is defined, they can be put together to form a coupled model. A Cell-DEVS coupled model is defined by:

$$GCC = < X_{\mathbf{list}}, Y_{\mathbf{list}}, X, Y, n, \{t_1,...,t_n\}, N, C, B, Z > \quad (4)$$

The cell space **C** defined by this specification is a coupled model composed by an array of atomic cells with size $\{\mathbf{t_1\ x...x\ t_n}\}$. Each cell in the space is connected to the cells defined by the neighborhood **N**, and the border (**B**) can have different behavior. The **Z** function allows one to define the internal and external coupling of cells in the model. This function translates the outputs of output port m in cell $\mathbf{C_{ij}}$ into values for the m input port of cell $\mathbf{C_{kl}}$. The input/output coupling lists (**Xlist**, **Ylist**) can be used to interchange data with other models.

The CD++ tool [6] 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 the size of the cell-space, the type of borders, a cell's interface with other DEVS models and a cell's behavior. The behavior of a cell (the $\boldsymbol{\tau}$ function of the formal specification) is defined using a set of rules of the form: *VALUE DELAY CONDITION*. When an external event is received, the rule evaluation process is triggered to calculate the new cell value. Starting with the first rule, the CONDITION is evaluated. If it is satisfied, the new cell state is obtained by evaluating the VALUE expression. The cell will transmit these changes after a DELAY. If the condition is not valid, the next rule is evaluated repeating this process until a rule is satisfied.

The specification language has a large collection of functions and operators. The most common operators are included: boolean, comparison, and arithmetic. In addition, different types of functions are available: trigonometric, roots, power, rounding and truncation, module, logarithm, absolute value, minimum, maximum, G.C.D. and L.C.M. Other available functions allow checking if a number is integer, even, odd or prime. In addition, some common constants are defined.

We will show how to apply Cell-DEVS to simulate different systems. We describe different models that are implemented and executed using the CD++ tool, fo-

cusing on particular characteristics of each system. We will show how complex applications can be implemented in a simple fashion.

A model of wave propagation

The state of a wave on water is characterized by its phase, intensity, direction and frequency. We built a model of wave propagation, which is based on sine waves [7]. If two waves with the same frequency are combined, there is a constant interference pattern caused by their superposition. Interference can be either constructive (meaning that the strength increases), or destructive (strength is reduced). In destructive interference, after superposition, the resultant wave will correspond to the sum of both waves. As phases are different, the wave will be canceled in those zones where there is overlap. The amount of interference depends of the phase difference in a point.

When a wave encounters a change in the medium, some or all the changes can propagate into the new medium (or it can be reflected from it). The part that enters the new medium is called the transmitted portion, and the other the reflected portion. The reflected portion depends on the characteristic of the incident medium: if this has a lower index of refraction, the reflected wave has an 180˚ phase shift upon reflection. Conversely, if the incident medium has a larger index of refraction, the reflected wave has no phase shift. In order to simulate the interference between waves and the propagation in CD++, we defined a multidimensional model, in which each plane was defined by every direction of wave spread (four directions for this example). We defined an integration plane, which is a composition of the direction planes, and contains the value or intensity of the wave corresponding to this position. Every cell in the cell space represents the minimal possible portion of the medium in which the wave propagates. Each cell has two values, phase and intensity. An integer value between zero and eight represent phase and a fractional value between zero and one (intensity).

```
rule: {0} 100 {trunc(0,0,0)=#maxFase or (fractional(0,0,0)<.0001 and
            fractional((0,0,0))>0)}
rule: {trunc(1,0,0)+fractional(1,0,0)*#attenuation} 100 {(1,0,0)!=0}
rule: {trunc(0,0,0)+1+fractional(0,0,0)} 100 { (0,0,0)!=0}
```

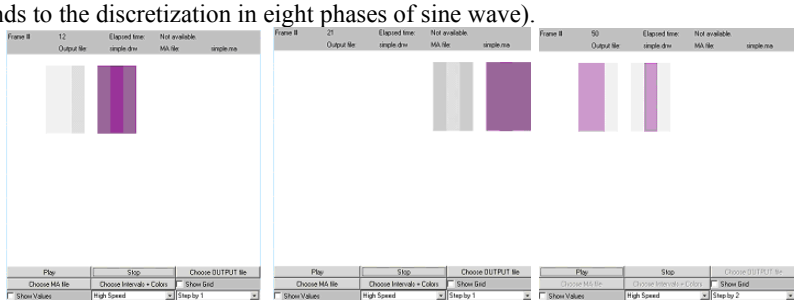
Fig. 1. Rules for wave propagation.

Figure 1 shows the rules used in CD ++ to specify the spread behavior. The first rule governs the attenuation of the wave. If the wave intensity is below of 0.0001 the wave propagation stops. The second rule contemplates the spread of the wave towards its neighbors, which preserve the phase of the wave but attenuate the intensity. The third rule contemplates the spread of the wave in the current cell. In this case, the cell intensity value does not change (only the phase).

```
rule: {((sin(PI/4*trunc((0,0,1))) * fractional((0,0,1)) + sin(PI/4 *
trunc((0,0,2))) * fractional((0,0,2)) + sin(PI/4 * trunc((0,0,3))) *
fractional((0,0,3)) + sin(PI/4*trunc((0,0,4))) * fractional((0,0,4))
)) * 10 } 100 {(0,0,1)!=0 or (0,0,2)!=0 or (0,0,3)!=0 or (0,0,4)!=0}
```

Fig. 2. Integration Rule.

Figure 2 shows the rule describing the values of the direction planes used in order to obtain the wave value in the medium in which is traveling (the value corresponds to the discretization in eight phases of sine wave).



(a) **(b)** **(c)**

Fig. 3. Result of wave propagation. (a) A wave traveling from left to right. (b) The wave reflects with the right border. (c) The wave before reflecting with the left border.

Figure 3 shows the simulation results of the wave model. Only the integration plane is showed (direction and intensity). It is possible to appreciate that the wave propagating produce attenuation.

Predator-Prey model

In this model, a predator seeks a prey, which tries to escape [8]. With predators always searching, prey must constantly avoid being eaten. A predator detects prey by smelling, as prey leave their odor when moving. The predator moves faster than the prey, and when there is a prey nearby, it senses the smell, and moves silently towards the prey. Each cell in the space represents an area of land (with vegetation, trees, etc.). Dense vegetation does not allow animals to advance, and thick vegetation form a labyrinth. When a prey is in a cell, it leaves a track of smell, which can be detected for a specific period (depending on weather conditions).

The cell's states are summarized in the following table. For example, the cell value 214 represents a prey following vegetation to find the exit in E direction.

Table 1. The following table describes the cell state codification using a natural number

Cell Value	Description
1..4	An animal moving towards N (1), W (2), S (3) or E (4).
200	Prey looking for the labyrinth's exit.
210	Prey following the forest to find the exit.
300	Predator looking for the labyrinth's exit.
310	Predator following the forest to find the exit.

400	Thick forest (does not allow animal movement).
0	Thin forest (allows animal movement).
101..104	Smell.

Figure 4 shows the specification of the model using CD++.

```
[pred-prey]
type : cell          dim : (20,20)        delay : inertial
neighbors: (-1,-1)(-1,0) (-1,1) (0,-1) (0,0) (0,1) (1,-1) (1,0) (1,1)
border : nowrapped       localtransition : movement

[movement]
rule : { (0,0) } 100000 { (0,0)=400}

%Predator moving towards N
rule: 0 800 {(-1,0)>100 and (-1,0)<250 and ((0,0)=311 or (0,0)=301)}
rule: 311 800 {(0,0)>100 and (0,0)<250 and ((1,0)=311 or (1,0)=301)}

%Predator moving towards East
rule: 0 800 {(0,1)>100 and (0,1)<250 and ((0,0)=313 or (0,0)=303)}
rule: 313 800 {(0,0)>100 and (0,0)<250 and((0,-1)=311 or (0,-1)=301)}
...
%Rules from smell path
rule: {(0,0)-1} 1000 {(0,0)>101 and (0,0)<105}
rule: 0 1000 {(0,0)<=101}
rule: {(0,0)} 100 {t}
```

Fig. 4. Specification of prey-predator's model.

In Table 1, cell's value equals to 400 represents forest where the prey and predator can't move (labyrinth). To specify this cell behavior we use a special rule (first rule in Figure 4), which is evaluated if only if the cell's value is equal to 400. The second and third rules govern the predator movement towards N. In this model the cell's value from 200, 210, 300 and 310 finished in 1 represents *toward N*, finished in 2 represents *toward East*, finished in 3 represents *toward S* and finally cell's value finished in 4 represents *toward W*. So, as you can see in the second rule, a cell's value equals to 301 o 311 represents a predator following the labyrinth toward N. In this model, the movements rules are in pairs, because one rule is used to move the prey o predator to the new position and the other is used to actualize the cell where the animals was. In the movement rules we use a 800 msec delay, which represents the speed of the predator moving in the forest. The last 2 rules represent the smell path for a prey. As weather conditions disperse the smells, so we use four different values to represent different dispersion phases. The first line in the *smell* rules govern the smell attenuation by subtracting 1 to the actual cell's value every second. The last rule change the actual cell's value to zero (no smell in the cell).

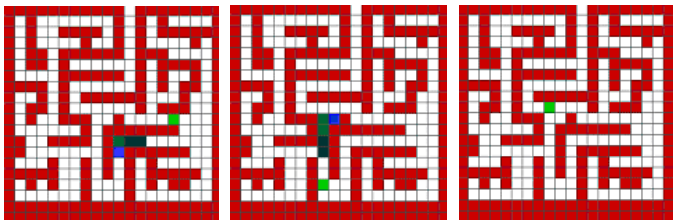


Fig. 5. Execution result of prey-predator's model [8].

Figure 5 shows the execution results of the model. A prey is trying to escape from the predator. Finally, the predator eats the prey.

Evacuation processes model

The simulation of evacuation processes has originally been applied to buildings or the aviation industry. Recently, however, the models have been transferred to ship evacuation taking into account special circumstances [9,10,11]. Our model represents people moving through a room or group of rooms, trying to gather their belongings or related persons and to get out through an exit door. The goal is to understand where the bottlenecks can occur, and which solutions are effective to prevent congestion [12].

The basic idea was to simulate the behavior and movement of every single person involved in the evacuation process. A Cell-DEVS model was chosen with a minimum set of rules to characterize a person's behavior:

- A normal person goes to the closest exit.
- A person in panic goes in opposite direction to the exit.
- People move at different speeds.
- If the way is blocked, people can decide to move away and look for another way.

Figure 6 shows the main rules of evacuation model. We have to different planes to separate the rules that govern the people moving among walls or isles from the orientation guide to an exit.

```
[deck]
dim : (18,18,2)        delay : inertial       border : wrapped
localtransition : EvaRule
neighbors : (-1,-1,0) (-1,0,0) (-1,1,0) (0,-1,0) (0,0,0) (0,1,0)
(1,-1,0) (1,0,0) (1,1,0) (-1,-1,1) (-1,0,1) (-1,1,1) (0,-1,1) ...

[EvaRule]
% Rules to govern people movement
rule : {trunc((0,0,0)/10)*10+1} {1000 / remainder(trunc((0,0,0)
/10),10) } {((0,0,0)>0 AND  remainder(trunc((0,0,0)/1),10) =0 AND
remainder(trunc((0,0,0)/100000),10) =0 AND ((0,-1,0)=0 OR (0,-1,0)=-
2) AND cellPos(2)=0 )AND (((0,-1,1) <= (1,-1,1) OR (1,-1,0)>0 OR (1,-
1,0)=-1 OR (randint(5)=0) ) AND ((0,-1,1) <= (1,0,1) OR (1,0,0)>0 OR
(1,0,0)=-1 OR (randint(5)=0) ) AND ((0,-1,1) <= (1,1,1) OR (1,1,0)>0
OR (1,1,0)=-1 OR (randint(5)=0) ) AND ((0,-1,1) <= (0,1,1) OR
(0,1,0)>0 OR (0,1,0)=-1 OR (randint(5)=0) ) AND ((0,-1,1) <= (-1,1,1)
OR (-1,1,0)>0 OR (-1,1,0)=-1 OR (randint(5)=0) ) AND ((0,-1,1) <= (-
1,0,1) OR (-1,0,0)>0 OR (-1,0,0)=-1 OR (randint(5)=0) ) AND ((0,-1,1)
<= (-1,-1,1) OR (-1,-1,0)>0 OR (-1,-1,0)=-1 OR (randint(5)=0) ))}
...

% Rules to control panic behavior
rule : {trunc((0,0,0)/10)*10+1} {1000/remainder(trunc((0,0,0)/10),10)
} {((0,0,0)>0 AND remainder(trunc((0,0,0)/1),10)=0 AND remaind-
er(trunc((0,0,0)/100000),10)>0 AND ((0,-1,0)=0 OR (0,-1,0)=-2) AND
cellPos(2)=0)AND (((0,-1,1)>= 1,-1,1) OR (1,-1,0)>0 OR (1,-1,0)=-1)
AND ((0,-1,1)>=(1,0,1) OR (1,0,0)>0 OR (1,0,0)=-1) AND ((0,-1,1)>=
```

```
(1,1,1) OR (1,1,0)>0 OR (1,1,0)=-1) AND ((0,-1,1)>=(0,1,1) OR
(0,1,0)>0 OR (0,1,0)=-1) AND ((0,-1,1)>=(-1,1,1) OR (-1,1,0)>0 OR (-
1,1,0)=-1) AND ((0,-1,1)>=(-1,0,1) OR (-1,0,0)>0 OR (-1,0,0)=-1) AND
((0,-1,1)>=(-1,-1,1) OR (-1,-1,0)>0 OR (-1,-1,0)=-1))}
```

Fig. 6. Specification of evacuation model.

The following table describes the encoding of the cell state, in which each position of the state is represented by natural number, in which each digit represent:

Digit	Meaning
6	Next movement direction. 1:W; 2:SW; 3:S; 4:SE; 5:E; 6:NE; 7:N; 8:NW
5	Speed (cells per second: 1 to 5)
4	Last movement direction, it can vary from 1 to 8 (as digit #6)
3	Emotional state: the higher this value is the lower the probability that a person gets in panic and find a wrong path.
2	Number of Movement that increase the potential of a cell
1	Panic Level, represent the number of cells that a person will move increasing the cell potential.

We used two planes: one for the floor plan of the structure and the people moving, and the other for orientation to an exit. Each cell in the grid represents 0.4 m^2 (one person per cell). The orientation layer contains information that serves to guide persons towards emergency exits. We assigned a potential distance to an exit to every cell of this layer. The persons will move for the room trying to minimize the potential of the cell in which they are (see Figure 6)

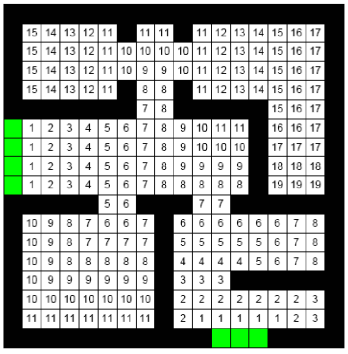


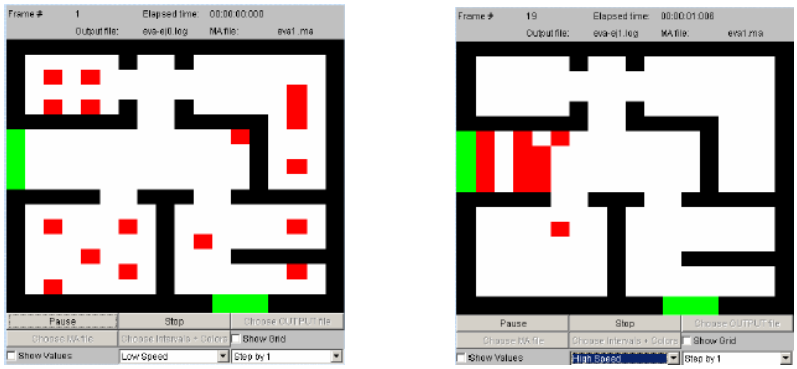
Fig. 7. Orientation layer: potential value (people try to use a path decreasing the potential).

The first set of rules in Figure 8 serves to define what path a person should follow using the orientation plane. The basic idea is to take the direction that decrease the potential of a cell, building a path following the lower value of the neighbors. We used the *remainder* and *trunc* functions to split the different parts of a cell's value. Also, we use the CD++ function *randint* to generate integer random values. We have 8 rules to control the people's movement, one for each direction. In all cases the rule

analyze the 8 near neighbors to understand what direction the person should take. We use *randint* for the case that all the 8 near neighbors has the same value.

The second set of rules governs the panic behavior: a person will take a wrong path or will not follow the orientation path. In that case, the direction will be calculated taking the path where the cell's potential is increased. In this case also we analyze the 8 near neghibors. This model doesn't allow people collitions, so every time that a person move, the destination cell must be empty.

Figure 7 shows the simulation results. The gray cells represent people who want to escape using the exit doors. The black cells represent walls. Note that the leftmost part in the figure shows people waiting in the exit door.



(a) (b)

Fig. 8. (a) People seeking an exit. (b) after 15 seconds, people found the exit.

Flock of birds

The motion of a flock of birds seems fluid, because of the interaction between the behaviors of individual birds. To simulate a flock we simulate the behavior of an individual bird (or at lE that portion of the bird's behavior that allows it to participate in a flock), based on the following behavior rules [13]:

- Collision Avoidance: avoid collisions with nearby flock mates.
- Velocity Matching: attempt to match velocity with nearby flock mates.
- Flock Centering: attempt to stay close to nearby flock mates

Characteristics of the birds:

- The birds fly in certain direction at a certain speed.
- The field of vision of the birds is 300 grades, but only they have good sight forward (in a zone from 10 to 15 grades)

Based on those rules we built a Cell-DEVS model to simulate the birds' fly [14]. Each cell of the model represents a space of 4 m^2, which can fit a medium size bird (~18-23 cm.). A second of simulation's time represents a second of real time. Therefore, a bird that moves in the simulation with a speed of 7 cells per second,

represents to a bird that flies to 14 m/s. The cell state codification represents with a natural number the direction of the bird (1:NW; 2:N; 3:NE; 4:W; 6:E; 7:SW; 8:S; 9:SE) and the speed. For example, the cell value 10004 represents a bird flying to W (unit value equal to 4) at 1 second per cell.

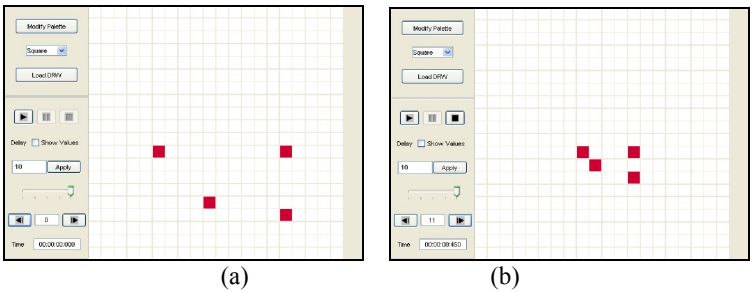
To avoid the collision of birds, when two or more birds want to move to the same place, they change their direction using a random speed variable. Figure 10 describes the model specification in CD++.

```
[boids]
type : cell        dim : (20,20)     delay :  transport
border : wrapped
neighbors : (-2,-2) (-2,-1) (-2,0) (-2,1) (-2,2) (-1,-2) (-1,-1) (-
1,0) (-1,1) (-1,2) (0,-2) (0,-1) (0,0) (0,1) (0,2) (1,-2) (1,-1)
(1,0) (1,1) (1,2) (2,-2) (2,-1) (2,0) (2,1) (2,2)
...
[fly-rule]
rule : { 1+if(((-2,-2)>100000),1,0)+if(((-2,-1)>100000),1,0)+
if(((-2,0)>100000),1,0)+if(((-2,1)>100000),1,0)+
if(((-2,2)>100000),1,0)+if(((-1,-2)>100000),1,0)+
if(((-1,-1)>100000),1,0)+if(((-1,0)>100000),1,0)+
if(((-1,1)>100000),1,0)+if(((-1,2)>100000),1,0)+
if(((0,-2)>100000),1,0)+if(((0,-1)>100000),1,0)+
if(((0,1)>100000),1,0)+if(((0,2)>100000),1,0)+
if(((1,-2)>100000),1,0)+if(((1,-1)>100000),1,0)+
if(((1,0)>100000),1,0)+if(((1,1)>100000),1,0)+if(((1,2)>100000),1,0)+
if(((2,-2)>100000),1,0)+if(((2,-1)>100000),1,0)+
if(((2,0)>100000),1,0)+if(((2,1)>100000),1,0)+if(((2,2)>100000),1,0)}
{90+trunc((0,0)/10-10000)*10}  {(0,0)>100000  AND  (((-1,-1)>100  AND
(-1,-1)=9100)  OR  ((-1,0)>100  AND  (-1,0)=8100)  OR  ((-1,1)>100  AND
(-1,1)=7100)  OR  ((0,-1)>100  AND  (0,-1)=6100)  OR  ((0,1)>100  AND
(0,1)=4100)  OR  ((1,-1)>100  AND  (1,-1)=3100)  OR  ((1,0)>100
AND (1,0)=2100) OR ((1,1)>100 AND (1,1)=1100))}
...
```

Fig. 10. Specification of the Flock of birds model.

Figure 10 shows the execution of the model using CD++. Basically the rules represents the fly behavior. Birds fly in a freedom way, instinctively when a bird detect others birds, try to follow them. The bird change the direction and the velocity to avoid collitions or lost the flock. In this model, we using different time conditions to simulate the change of bird's velocity.

We show the birds flying, and when one bird sees the other, they start flying together.



(a) (b)

Fig. 11. Joining behavior (a) four birds flying isolated; (b) birds flying together.

Conclusion

Cell–DEVS allows describing physical and natural systems using an n-dimensional cell-based formalism. Input/output port definitions allow defining multiple interconnections between Cell-DEVS and DEVS models. 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 also can develop multidimensional models, making the tool a general framework to define and simulate complex generic models.

The tool and the examples are the public domain and they can be obtained in:
`http://www.sce.carleton.ca/faculty/wainer/`

References

1. M. Sipper. "The emergence of cellular computing". IEEE Computer. July 1999. Pp. 18-26.
2. D. Talia. "Cellular processing tools for high-performance simulation". IEEE Computer. September 2000. Pp. 44 –52.
3. S. Wolfram. “Theory and applications of cellular automata”. Vol. 1, Advances Series on Complex Systems. World Scientific, Singapore, 1986.
4. G. Wainer; N. Giambiasi,. "Timed Cell-DEVS: modelling and simulation of cell spaces". In Discrete Event Modeling & Simulation: Enabling Future Technologies. 2000. Springer-Verlag.
5. B. Zeigler, H. Praehofer, T. Kim. Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. 2000. Academic Press.
6. Wainer, G. "CD++: a toolkit to define discrete-event models". Software, Practice and Experience. Wiley. Vol. 32, No.3. pp. 1261-1306. November 2002.
7. D’Abreu, M.; Dau, A.; Ameghino, J. "A wave collision model using Cell-DEVS". Internal report, Computer Science Department. Universidad de Buenos Aires. 2002.
8. Baranek, A.; Riccillo, M.; Ameghino, J. "Modelling prey-predator using Cell-DEVS". Internal report, Computer Science Department. Universidad de Buenos Aires. 2002.
9. Klüpfel, H.; Meyer-König, T.; Wahle J.; Schreckenberg, M. “Microscopic Simulation of Evacuation Process on Passenger Ships”. In "Theoretical and Practical Issues on Cellular Automata”, Springer-verlag 2001.
10. Meyer-König, T.; Klüpfel, H.; Schreckenberg, M. “A microscopic model for simulating mustering and evacuation processes onboard passenger ships”. In Proceedings of TIEMS 2001, Oslo, Norway. 2001.
11. Hongtae Kim, Dongkon Lee, Jin-Hyoung Park, Jong-Gap Lee, Beom-Jim Park and Seung-Hyun Lee. “Establishing the Methodologies for Human Evacuation Simulation in Marine Accidents”. Proceedings of 29th Conference on Computers and Industrial Engineering. Montréal, QC. Canada. 2001.
12. Brunstein, M; Ameghino, J. “Modeling evacuation processes using Cell-DEVS”. Internal report. Computer Science Department. Universidad de Buenos Aires. 2003.
13. Reynolds Craig, W. “Flocks, Herds, and Schools: A Distributed Behavioral Model”. Computer Graphics 21(4), pp. 25-34, July 1987.
14. Dellasopa J; Ameghino, J. “Modelling a Flock of birds using Cell-DEVS”. Internal report. Computer Science Department. Universidad de Buenos Aires. 2003.