A Model of Virus Spreading Using Cell-DEVS

Hui Shang and Gabriel Wainer

Department of Systems and Computer Engineering, Carleton University,
1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6 Canada
{shanghui, gwainer}@sce.carleton.ca

Abstract. Cell-DEVS is a combination of CA with the DEVS formalism that allows the definition of cellular models. CD++ is a modeling and simulation tool that implements DEVS and Cell-DEVS. We have used CD++ to build a model about competition between population and viruses. We will discuss how to define such a model in CD++, and will show simulation results under different scenarios.

1 Introduction

In recent years, many simulation models of real systems have been represented using Cellular Automata (CA) [1]. CA are defined as infinite n-dimensional lattices of cells whose values are updated according to local rules. Cell-DEVS [2] was defined as a combination of CA and DEVS (Discrete Events Systems specifications) [3]. A Cell-DEVS model is defined as a lattice of cells holding a state variable and a computing apparatus to update the cell state. Each cell in a Cell-DEVS is a DEVS atomic model, and the cell space is a DEVS coupled model. Once the cell behavior is defined, a coupled Cell-DEVS is created by putting together a number of cells interconnected by a neighborhood relationship. CD++ [4] is a simulation tool based on Cell-DEVS. A built-in specification language provides a set of primitives to define the cell spaces. We have used CD++ to build a Cell-DEVS model on competition between population and viruses, based on the work presented in [5]. The model describes evolution of a population and the interaction between individuals and viruses. Cells valued 1-6 represent individuals (1-young; 2-5: mature; 6: aged). Individuals in cells will use the following rules:

1. *Age increment:* periodically, each cell will be incremented to indicate aging. After reaching the maximum age (6), individual dies (0).
2. *Reproduction:* for each unoccupied cell with at least two adult neighbors (Von Neumann's neighborhood), the cell is set to one.
3. *Movement:* mature individuals can move at random.

Viruses are represented by cells valued 8 (active) or 9 (inactive). They comply with the following rules:

1. *Virus reproduction:* when an unoccupied cell is surrounded by at least one active virus, after a delay, an active virus will occupy the cell.
2. *Virus state change:* after a delay, active virus will become inactive (from 8 to 9). After another delay, the inactive virus will die (from 9 to 0).

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 373 – 377, 2005.

Individuals and viruses compete for the living space as follows:

1. *Virus killing individuals: if* an individual is surrounded by at least two viruses with vertical or horizontal distribution, the individual and the viruses will die.
2. *Individuals killing virus:* if an active virus is surrounded by at least two individuals, and the viruses have no capacity to kill individuals, the virus dies.
3. *Conflict resolution:* the following conflicts are handled:

- **Empty cell occupation:** an empty cell may be occupied by moving individuals, newborn individuals or active viruses. We give the highest priority to movement, then to reproduction, and lowest priority to virus reproduction.
- **Individual/virus interaction:** if individuals move, they might become out of the range of the virus. In this case, the movement has higher priority.
- **Individual reproduction/movement:** reproduction and movement can happen simultaneously. In that case, the empty cell becomes a newborn individual, while its parents move to other places.

2 Model Execution

The model was implemented in CD++, and a set of experiments was carried out with six different settings. A detailed model definition can be found in [6], and an excerpt can be seen in the Appendix. In the model definition, each cell is associated with cell I/O ports. One of them (*pa*) represents the age of individuals and virus. The other (*pd*) represents the direction of moving individuals. Execution results are shown for different scenarios (gray cells change from light to dark to indicate age; darker cells present active and inactive viruses).

Scenario 1. Population is partially congregated, and viruses are scattered inside the population. No movement rules are applied.

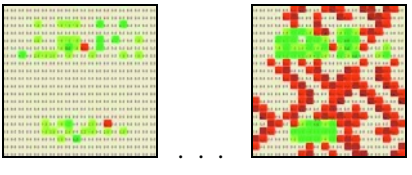

Fig. 1. Virus spread scenario

When compared with the size of population, the number of viruses increases, reflecting the reproduction rules (individuals have more strict rules than viruses). The distribution of population allows individuals to reproduce; nevertheless, the population cannot develop properly due to the action of viruses. The partial congregation of population also provides some space for virus reproduction, so viruses can reproduce quickly. Since the conflict rules give higher priority to population over viruses, it there is a tendency for individuals to congregate.

Scenario 2. Population is packed; viruses are scattered. No movement rules.

The total congregation can prevent the individuals from being killed by viruses, however, it also restrict reproduction. Since viruses scatter inside the population, their reproduction is also restricted. The population size grows, while viruses disappear.

Fig. 2. Congregated population scenario

Since the individuals separate, there is more space to reproduce. However, the number of individuals decreases due to the development of viruses. The number of individuals who survive finally decreases when compared with previous examples:

- population tends to congregate. Reproduction leads to congregation, and congregation is helpful to avoid being killed. However, the introduction of movement has an opposite effect.
- as movement rules have higher priority than reproduction, the possibilities to reproduce are smaller.
- In the previous examples, the initial distribution contained more young individuals. Here, the age is uniformly distributed, and elder individuals die earlier.

Scenario 3. Individuals disperse, viruses scattered. Movement rules are applied.

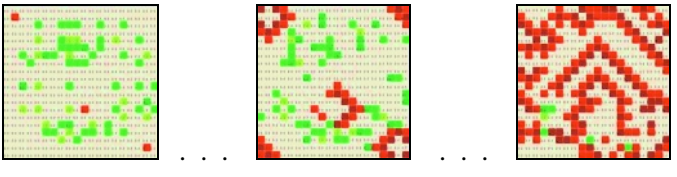

Fig. 3. Movement scenario

3 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 model on competition between population and viruses. We showed how to define such a model in CD++, and presented different simulation results under different

scenarios. We extended the basic behavior of such model including mobility, showing how to define such a model using CD++ specification facilities.

References

1. Wolfram, S. 2002. A new kind of science. Wolfram Media, Inc.
2. G. Wainer, N. Giambiasi: N-dimensional Cell-DEVS. Discrete Events Systems: Theory and Applications, Kluwer, Vol.12. No.1 (January 2002) 135-157
3. B. Zeigler, T. Kim, H. Praehofer: Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press. 2000.
4. G. Wainer: CD++: a toolkit to define discrete-event models. Software, Practice and Experience. Wiley. Vol. 32. No.3. (November 2002) 1261-1306
5. A Cellular Automata Model of Population Infected by Periodic Plague. P.M.A. Sloot, B. Chopard, and A.G. Hoekstra (Eds.): ACRI 2004, LNCS 3305, pp. 464–473, 2004.
6. H. Shang, G. Wainer. A model of virus spreading in CD++. Technical Report SCE-05-05. Systems and Computer Engineering. Carleton University. 2005.

Appendix (Excerpt of the Update Rules for the Model)

```
rule : {~pa:=(0,0)~pa+1[1]; ~pd:=round(uniform[2](1,4));}[3] 100[4]  { (0,0)~pa
=1 and (not ((0,-1)[5]~pa=8 and (0,1)~pa=8)) and not ((-1,0)~pa = 8 and
(1,0)~pa = 8)}[6]  %Age increment for newborns not killed by viruses. rule
: {~pa:=0;} 100 {(0,0)~pa=1 and (((0,-1)~pa=8 and (0,1)~pa=8) or ((-
1,0)~pa=8 and (1,0)~pa=8))}   ; virus killing individuals
%Moving rules for mature cells.
rule : {~pa:=0;} 100 {(0,0)~pa>=2 and (0,0)~pa<=5 and (0,0)~pd=4 and
(0,-1)~pa = 0}
  ...
rule:  {~pa:=0;} 100 {(0,0)~pa=6} % Dying individual
rule: {~pa:=0;} 100 {(0,0)~pa=8 and (((0,-1)~pa>=1 and (0,-1)~pa<=6) or
((0,1)~pa>=1 and (0,1)~pa<=6) or ((-1,0)~pa>=1 and (-1,0)~pa<=6) or
((1,0)~pa>=1 and (1,0)~pa<=6))} %Individual kill virus rule
rule:   {~pa:=9;} 100 {(0,0)~pa=8 and not ( ((0,-1)~pa>=1 and (0,-
1)~pa<=6) or ((0,1)~pa>=1 and (0,1)~pa<=6) or ((-1,0)~pa>=1 and (-
1,0)~pa<=6) or ((1,0)~pa>=1 and (1,0)~pa<=6))}
...
rule: {~pa:=0;} 100 {(0,0)~pa=9}
rule: {~pa:=(0,1)~pa+1; ~pd:=round(uniform(1,4));} 100 {(0,0)~pa=0 and
((0,1)~pa>=2 and (0,1)~pa<=5) and (0,1)~pd=4} % Virus: active to passive
%Individual reproduction rule
rule: {~pa:=1;} 100 {(0,0)~pa=0 and ( ((0,1)~pd !=4 or (0,1)~pa<2 or
(0,1)~pa>5) and ((-1,0)~pd !=3 or (-1,0)~pa<2 or (-1,0)~pa>5) and ((0,-
1)~pd !=2 or (0,-1)~pa<2 or (0,-1)~pa>5) and ((1,0)~pd !=1 or (1,0)~pa<2
or (1,0)~pa>5) ) and ((((0,-1)~pa>=2 and (0,-1)~pa<=5) and ((0,1)~pa>=2
```

[1] (0,0)~pa: (0,0) : cell reference, ~pa: associated port.
[2] Random number generator using Uniform distribution.
[3] Postcondition: the corresponding port of reference cell will be updated according this.
[4] Delay time. After the proposed delay time, the output port value of each cell will be updated.
[5] Relative position to the reference cell.
[6] Precondition part. Used to evaluate the current reference cell, if the condition is true, then the cell will be valued according to the value part.

```
and (0,1)~pa<=5) ) or ( ((-1,0)~pa>=2 and (-1,0)~pa<=5) and ((1,0)~pa>=2
and (1,0)~pa<=5)))}
%Virus reproduction rule
rule: {~pa:=8;} 100 {(0,0)~pa=0 and ( ((0,1)~pd !=4 or (0,1)~pa<2 or
(0,1)~pa>5) and ((-1,0)~pd !=3 or (-1,0)~pa<2 or (-1,0)~pa>5) and ((0,-
1)~pd !=2 or (0,-1)~pa<2 or (0,-1)~pa>5) and ((1,0)~pd !=1 or (1,0)~pa<2
or (1,0)~pa>5) ) and not((((0,-1)~pa>=2 and (0,-1)~pa<=5) and
((0,1)~pa>=2 and (0,1)~pa<=5)) or (((-1,0)~pa>=2 and (-1,0)~pa<=5) and
((1,0)~pa>=2 and (1,0)~pa<=5))) and (((0,-1)~pa=8 or (0,1)~pa=8) or ((-
1,0)~pa=8 or (1,0)~pa=8))}
```