Forest fire modelling and simulation in the DELTA environment

Fernando J. Barros *, Maria T. Mendes

Departamento de Engenharia Informática, Universidade de Coimbra, Pólo II, P-3030 Coimbra, Portugal

Received 6 October 1995; revision 4 March 1996

Abstract

Traditional simulation methodology supports only changes in models state variables. Some models are better expressed by a combination of both changes in state variables and changes in structure. Dynamic Structure Discrete Event Specification (DSDEVS) is a recently introduced modelling and simulation formalism that provides full support for representing models with time varying structure. The DELTA simulation environment is an implementation of the DSDEVS formalism and provides full support to Structural Simulation. We show the advantages of the Dynamic Structure Cellular Automata describing the model of forest fire spreading and its implementation in the DELTA environment. © 1997 Published by Elsevier Science B.V.

Keywords: Modelling methodology; DSDEVS formalism; Dynamic structure cellular automata; Fire modelling

1. Introduction

Traditional simulation methodologies provide support only for changes in model descriptive variables. Many systems are better represented by models with both changes in state variables and changes in their structure. Among these systems for example are adaptive computer architectures, fault-tolerant networks, self-reconfiguring manufacturing systems and self-organizing systems. Although changes in structure can be represented to a certain extent in conventional simulation formalisms they do not provide full and sounds semantics to accomplish this task. These formalisms force the representation of structural changes as simple state changes. The Dynamic Structure Discrete Even System Specification (DSDEVS) has been introduced in [1–3] and provides full support for Structural Simulation. Structural
simulation as opposed to conventional *Trajectory Simulation* concerns models that change their structure dynamically. A dynamic structure model has the ability to transform itself into another model of a variant family. The DSDEVS formalism provides complete semantics to modelling and simulation of dynamic structure systems. DSDEVS supports changes ranging from simple linking relations between models, to more complex changes in model composition such as adding and deleting models during simulation. Exchanging models between networks can be easily accomplished in the DSDEVS formalism. Related work on dynamic structure modelling and simulation can be found in [6, 10, 11, 14, 15].

*Cellular Automata* (CA) have been widely used for fire modelling [5, 11]. Although cellular automata have a great potential for fire modelling, their use in large problems can be impossible due to the number of cells required. The size of conventional CA is fixed and cannot change during simulation. Even if the simulation activity is limited to a small part of the model the entire set of cells must be present. To overcome these problems we use *Dynamic Structure Cellular Automata* (DSCA) to represent only the active model cells. DSCA can change its own structure and thus can keep just the cells needed to represent the modeled system. The DSCA use in fire modelling increases the size of the model that can be represented by keeping just the active burning cells. We built a DSCA using the DELTA modelling and simulation environment [5].

Hierarchy and modularity have been recognized as key factors for building reusable and testable simulation models. DELTA provides an environment for hierarchical modular simulation. Its characteristics for dynamic structure modelling offer all the features needed for the construction of dynamic cellular automata. We present examples of model building with DELTA environment and its potential in reducing problem size. DELTA is implemented in Smalltalk language and runs on MS/Windows [8].

2. The DSDEVS formalism

The Dynamic Structure Discrete Event System Specification formalism (DSDEVS) is a new formalism to specify system that can change their structure dynamically. In the DSDEVS formalism there are two types of models, basic models and network models.

2.1. The DSDEVS basic model

Basic models are defined in the DSDEVS formalism, as in the DEVS formalism [12, 13], by the structure

\[ M = \langle X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, \tau \rangle \]

where

- \( X \equiv \text{set of input events}, \)
- \( S \equiv \text{set of sequential states}, \)
\[ Y \equiv \text{set of output events}, \]
\[ \delta_{\text{int}} : S \rightarrow S \equiv \text{internal transition function}, \]
\[ \delta_{\text{ext}} : X \times Q \rightarrow S \equiv \text{external transition function}, \]
\[ Q = \{(s, e) \mid s \in S, 0 \leq e \leq \tau(s)\} \equiv \text{total state set}, \]
\[ e \equiv \text{time elapsed since last transition}, \]
\[ \lambda : S \rightarrow Y \equiv \text{output function}, \]
\[ \tau : S \rightarrow \mathbb{R}_0^+ \equiv \text{time advance function}. \]

When an external event arrives the external function changes the model's state. The time advance function is the time that a model can remain in the same state. After this period the model undergoes an internal transition. The output value is computed just before the internal transition is triggered. A complete description of the interpretation of basic models can be found in [12].

2.2. The DSDEVS network model

Network models are a combination of DSDEVS basic models. In contrast to other simulation formalisms, that supports only static structure models, the structure of the DSDEVS networks can be changed. The DSDEVS dynamic structure network is defined by the structure

\[ \text{DSDEVN}_N = \langle \chi, M_{\chi} \rangle, \]

where
\[ A \equiv \text{DSDEVS network name}, \]
\[ \chi \equiv \text{DSDEVS network executive}, \]
\[ M_{\chi} \equiv \text{model of } \chi. \]

The DSDEVS network is defined with a special component, the network executive \( \chi \). \( M_{\chi} \), the model of the executive, is a DSDEVS basic model and is defined by the structure

\[ M_{\chi} = \langle X_{\chi}, S_{\chi}, Y_{\chi}, \delta_{\text{int}_{\chi}}, \delta_{\text{ext}_{\chi}}, \lambda_{\chi}, \tau_{\chi} \rangle. \]

The information about the dynamic structure network is located in the state of the executive. The executive, besides domain dependent state knowledge, holds the information about network composition and coupling. Changes in executive network related state variables will be automatically mapped onto changes in network structure

\[ s_{\chi} = \langle X_{\chi}', Y_{\chi}', D', \{M_{\chi}^i\}, \{I_{\chi}^i\}, \{Z_{\chi}^i\}, \Xi', \Theta' \rangle, \]

where
\[ X_{\chi}' \equiv \text{input event set of DSDEVS network}, \]
\[ Y_{\chi}' \equiv \text{output event set of DSDEVS network}, \]
\[ D' \equiv \text{set of components}, \]
\[ M_{\chi}^i \equiv \text{model of component } i, \text{ for all } i \in D', \]
\[ I_{\chi}^i \equiv \text{influences of } i, \text{ for all } i \in D' \cup \{\chi, A\}. \]
The state variables are subject to the following constraints:

\[ Z_{t,j} = \text{i-to-j translation function, for all } j \in I^j, \]
\[ \Xi^x = \text{select function}, \]
\[ \theta^x = \text{other state variables not defined before}. \]

The tuple \((X_3, Y_3, D^x, \{M_i\}, \{I^i\}, \{Z_{t,j}\}, \Xi^x)\) defined in the executive state is referred to here as the network structure. Any change in one of these variables is called a change in structure.
GIS data we calculate the fire Direction of Maximum Spread (DMS) and the fire Rate of Spread (ROS) calculated along the DMS.

3.1. The fire model

From the GIS data we calculate HROS and DMS. We use the cosine correction described in [7] to calculate the spread of the fire in each direction. This rate is given by

\[ ROS_{adj}(\gamma) = \frac{(1-\epsilon_l)ROS}{1-\epsilon_l \cos (\gamma - DMS)} \]

where

- \( ROS_{adj} \) = adjusted rate of spread [m/min],
- \( \gamma \) = direction of spread,
- \( ROS \) = heading rate of spread [m/min],
- \( DMS \) = direction of maximum spread,
- \( e_{wind} \) = effective wind speed [km/h],
- \( r_{lw} = 1 + 0.155 e_{wind} \) [m/min],
- \( \epsilon_l = \frac{dm}{r_{lw}} \).

Using these equations we calculate the adjusted rate of fire spread in each of the eight neighbors of the cell. The actual time for a cell to ignite its neighbor depends on the distance between the centers of the two cells.

3.2. The cell model

Each cell is a basic model and is represented by the structure

Cell = \( \langle S, Y, \delta_{int}, I, \tau \rangle \),

where \( s \in S \) is given by

\[ s = ((i, j), (e_1, \ldots, e_8), (d_1, \ldots, d_8), n), \]

\( (i, j) \in \mathbb{Z}^2 \), is the cell position,
\( e_1, \ldots, e_8 \in \mathbb{R}_0^+ \), are time intervals between successive neighbor ignition,
\( d_1, \ldots, d_8 \in \{N, NE, E, SE, S, SW, W, NW\} \), are the directions of fire propagation,
\( n \in \{1, \ldots, 10\} \), represents the cell current status.

The time advance function \( \tau \), is defined by

\[ \tau((i, j), (e_1, \ldots, e_8), (d_1, \ldots, d_8), n) = \begin{cases} e_n & \text{if } n \leq 8, \\ 0 & \text{if } n = 9, \\ \infty & \text{if } n = 10. \end{cases} \]
When the last neighbor was ignited the cell immediately undergoes an internal transition to send a remove command to the executive. After making this request the cell just passivates waiting to be removed.

The output set of a cell is given by

\[ Y \subseteq P \times \mathbb{I}^2 \times \{1, \ldots, 9\}, \]

where \( P \), the set of input ports, is given by

\[ P = \{ \text{spread}, \text{remove} \}. \]

Ports are used to structure the input set. Block diagrams and external functions make explicit use of port names.

The output function \( \lambda \), is given by

\[ \lambda(i, j), (e_1, \ldots, e_8), (d_1, \ldots, d_8), n) = \begin{cases} (\text{spread}, (i, j), d_n) & \text{if } n \leq 8, \\ (\text{remove}, (i, j), n) & \text{if } n = 9. \end{cases} \]

For \( n = 1, \ldots, 8 \) the cell just propagates the fire to its neighbors. When all the neighbors started burning, \( n = 9 \), the cell is no longer needed and it asks the executive to be removed.

The initial state \( s_0 \in S \) is given by

\[ s_0 = ((i, j), (e_1, \ldots, e_8), (d_1, \ldots, d_8), 1), \]

where the values \( e_1, \ldots, e_8 \) and \( d_1, \ldots, d_8 \), are computed using the equations described in the previous section. The value \( e_1 \) is the time to ignite the first neighbor cell and \( d_1 \) represents this cell direction.

The internal transition of a cell is given by

\[ \delta_{\text{int}}((i, j), (e_1, \ldots, e_8), (d_1, \ldots, d_8), n) = ((i, j), (e_1, \ldots, e_8), (d_1, \ldots, d_8), n + 1). \]

Just before the cell is removed its time advance function is \( \infty \), and as a result the internal function will not be triggered and no output will be produced.

Because a cell does not receive any input from other cells its model does not include neither the input set nor the external function.

3.3. The simulation model

The Systems Entity Structure (SES) was described in \([13, 16]\) and is a knowledge scheme to represent model composition and coupling. We use the SES to represent a fire model, Fig. 1. A single bar (|) represents single decomposition and a triple bar (|||) represents multi-decomposition. Network executives names are represented, in the SES, enclosed within a box. The model is decomposed in a Generator and a Dynamic Structure Cellular Automata (DSCA). A dynamic structure model has the ability to transform itself into a model in a variant family. In our case the SES will be used as the framework to represent a family of all cellular automata. Models can change their structure to another model in the variant family represented in the
SES. Changes in structure is represented by the multi-decomposition. The SES represents a cellular automata with an arbitrary number of cells.

The Generator element provides the data to initialize the forest model. With this separation between the model and the experimental frame we can study the same model under different conditions just by changing the Generator model. The DSCA has an arbitrary number of cells. Each cell models a small fraction of the forest and has information about fire spread characteristics. The DSDEVS formalism provides full support to structural changes with the help of the executive element DSCA.x.

This model keeps information about the active cells in the model and about the cells already burnt.

A cell is removed when it cannot ignite its neighbors. By changing the structure of the model we are able to greatly reduce the average number of cells.

To show the basic principles of the dynamic structure cellular automata we use a simple example of fire growth in homogeneous fuel conditions. Fire RMS is equal to 5 m/min. and each cell has 20 x 20 m. Fire will take 4 min to travel from the origin to the cells in North, South, East and West directions. The time the fire takes to reach the NE, SE, NW and SW cells is equal to 5.64 min.

The simulation starts with no cells in Fig. 2(a). At t = 0 one cell is ignited, as seen in Fig. 2(b). At this point there is just one active cell in the cellular automata. The simulation clock advances to the time of next event, t = 4 min. At this point North, South, East and West cells are created, (Fig. 2(c)). The first cell is still active because it can still start fire in other cells. The time of next event, when the remaining cells close to the starting cell are ignited, t = 5.64 min. At this moment NE, NW, SE and SW cells are created and the starting cell is removed because all its neighbors are already burning, Fig. 2(d). The number of cells at each moment is always a small fraction of the total number of cells of the overall model.

As seen in this example the DSCA has the ability to change their own structure and accommodate just the cells needed in the simulation process.

4. Implementation in DELTA environment

DELTA modelling and simulation environment [1] is an implementation of the DSDEVS formalism and provides full support to dynamic structure modelling and
simulation. The DSDEVS formalism supports model building in a hierarchical and modular manner. Models interaction is made through input and output ports.

The diagram block of the DSCA is represented in Fig. 3. Changes in structure are supported by the DSCA executive DSCA. This model keeps information about the DSCA composition and coupling.

The executive has six input ports. Ports **remove** and **spread** are used to remove a cell from simulation and to create more cells, respectively. Information about forest characteristic is given in port **data**. Fire starts in the cells received in **fire** port. Simulation results are written to a disk file when the executive receives a message in the **stop** port. Cell conditions can be changed during simulation by a message sent to **wind** port. This information is sent by the Generator model. To support changes in structure, internal and external transition functions can use the following methods to change the structure.

- **addModel**: *aModel*, adds a new model to the simulation;
- **removeModel**: *aModel*, removes a model from simulation. All links from and to the model are removed.
- **link**: *aModel port: aPort to: bModel port: bPort*, creates a link between two models.
An enumeration of executive methods is given in [1]. These methods provide sound operations to change the executive state variables and by consequence to change the network structure.

The executive keeps track of all burning cells in state variable burnt. This variable has also the time when each cell starts to burn. The translation between a new cell relative position to an absolute address is made by a mapping stored in state variable map.

Cells have the information about the time to ignite all their neighbors. When a cell propagates the fire to a neighbor it sends to the executive the new cell relative position. After the cell has ignited all the neighbors its role in the simulation is complete and the cell will be no longer needed. At this time the cell sends a message to the executive to remove it from the simulation.

The executive external function is responsible for adding new cells to the model and its realization is shown in Fig. 4.

When a cell cannot ignite more cells it asks the executive to be removed. The part of the external function responsible to delete a cell is represented in Fig. 5. When a model is removed all its connections are also deleted.

The executive starts the simulation when it receives a #fire command with the starting burning cells as arguments (Fig. 6). The time when a cell is created is stored for statistic analysis.

When a new cell is added, the links are created between the new cell and the executive. When a cell is removed all its links are automatically removed. Although we have constrained cell boundaries to lie in area limit, this restriction could be easily removed and we could simulate a model without any boundaries.
At the end of simulation, the time each cell started burning is written to a disk file. The separation between the model and the experimentation makes possible to use the same DSCA with different forest models. As it can be seen by this small example the potential for the reduction of the
number of cell models is very high. The DSCA represents models with a large number of cells by keeping only the active cells.

5. Simulation results

To assess simulator correctness we have tested it under simple, ideal conditions. We assumed homogeneous fuel type, no slope and a constant wind speed $\text{e}_{\text{wind}} = 6 \text{ km/h}$. We have used a $\text{HROS}$ of 10 m/min pointing North, and a cell size of 20 m. The burnt area after 70 min is represented in Fig. 7. As expected, the shape of the burnt area is an ellipse pointing to the wind direction.

Due to the discrete nature of cellular automata and to the simple fire model used, there is an error associated with fire prediction.

A conventional cellular automata will have a $50 \times 50 = 2500$ cells permanently. The results obtained in this simulation were,

Average number of cells = 105.9,
Maximum number of cells = 266.

These values are very low when compared with the number of cells in a CA. Although these figures depend on fire parameters we can see that a DSCA will keep only the burning cells which will be always a small fraction of the total number cells of the static automaton.

DSCA offer a potential for representing a virtually unbounded CA. This characteristic is of fundamental importance for modelling real forest fires. Even a small fire

Fig. 7. Simulation of fire spread with wind speed of 6 km/h.
in an area of $1000 \times 1000 = 10^6$ cells will be very difficult to model if represented by
conventional CA. DSCA offer a general framework to solve very large scale simulations.

The benefits of the DSCA will decrease if the time each cell remains in simulation
is very long. In this case the benefits of structural changes will be lost by the overhead
to perform add and remove operations. This kind of trade off seems similar to the
cost benefit relation in matrix representation. The use of a dynamic representation
for a cellular automata is a decision based on how sparse it is.

Dynamic structure methodology can, in general, be used to represent either models
that demand for large memory space or to make models easier to understand [4].
The time benefits of the dynamic structure approach depends on both the type of
model and on the efficiency of the implementation, and thus, no general conclusions
can be made. Time efficiency depends mainly on the algorithm used to handle link
information. By the contrary, space benefits and more faithful models do not depend
on the specific implementation, and results are valid for any realization of the
DSDEVS formalism.

In the DELTA environment the add and remove functions are time cost operations
and the further work is needed to make these operations more efficient. Another
improvement would involve the use of parallel computers. Nevertheless, the use of
the DSDEVS and the DELTA implementation to represent static models has no
overhead when compared, for example, with a similar implementation of the original
DEVs. In the former implementation coupling information is located in the executive
while in the later this information is stored in the coupled model. There is no
performance difference in these two different schemes.

6. Conclusions

Dynamic Structure Discrete Event System Specification provides full support for
dynamic structure modelling and simulation. Because DSDEVS supports hierarchical
and modular model building, DSDEVS formalism can be used to represent complex
models. Cellular automata have been widely used for forest fire modelling. Although
they offer a very powerful model, static cellular automata cannot be used in very
large problems. Dynamic structure cellular automata offer a good framework for
represent cellular automata with a very large number of cells. For large problems,
memory requirements of dynamic structure automata could be several orders of
magnitude lower than static automata. With this approach we can deal with essen-
tially unbounded problems. DELTA simulation environment, an implementation of
Dynamic Structure Discrete Event System Specification formalism, proved to be a
general framework for representing dynamic structure systems in general and
dynamic structure cellular automata in particular. As a continuation of the former
work we intend to extend DSDEVS formalism to exploit the parallelism existing in
the models and to improve the efficiency of the DELTA system.
Acknowledgements

The author wants to thank Prof. George Ball of the University of Arizona for his valuable suggestions. This work was supported by Comissão INVOTAN.

References