Closure under Coupling of Cellular-DEVS for the Optimization of Memory Resource: Wildfire Spread Case Study

Yousef Dahmani$^{1}$, Maamar El Amine Hamri$^{2}$, and Nesrine Driouche$^{3}$

$^{1}$Ibn Khaldoun University, Tiaret, Algeria (dahmani.youcef@yahoo.fr)
$^{2}$Aix-Marseille Univ, Universit de Toulon, CNRS LIS, Marseille, France (amine.hamri@lis-lab.fr)
$^{3}$Independant software researcher, Marseille, France (driouche.nesrine@hotmail.fr)

September 3, 2020

Abstract

The present work aims to show one of the advantages of using the property of closure under coupling in the DEVS specification. The advantage concerned in this paper attempts to address the need for memory resources during the simulation of systems by cellular-DEVS.

This improvement of performance is based on the usage of the property closure under coupling in the DEVS formalism. With this property and taking account of the iterative behavior of each cellular-DEVS atomic model, we provide simulation of many models simultaneously. The method starts with the specification of the cellular-DEVS coupled model which is then converted into its equivalent DEVS atomic model. Thus, the goal of this conversion is to transform large quantities of atomic models coupled together, which require huge computational resources, into one DEVS atomic model.

A case study is presented at the end of the work on modeling and simulation of forest fire propagation using DEVS and cellular-DEVS. A specification by cellular-DEVS of the forest fire model and its non-modular equivalent DEVS atomic model are presented. Finally, a comparison between both methods is presented in term of consumption of resources.

1 Introduction

Modeling and Simulation (M&S) have accumulated a large number of successes in a wide and varied range of domains. However, M&S consumes time, effort and resources when you need to resolve class of problems where the analytic or modeling solution is very hard to find. This class of problems includes dynamic systems where there is a wide number of parameters with time and causal dependencies between them, a non-linear behavior of the system, etc.

Discrete event M&S deals with systems whose temporal and spatial behaviors are complex to be treated analytically. The DEVS formalism (Discrete EVent system Specification) is one
of the common formalism used in the simulation of dynamic systems [1][2]. It is known for its modularity and expressiveness. This formalism offers, compared to others, a general purpose.

However, the DEVS formalism has undergone several extensions to meet specific needs. Many variants on DEVS were adopted by introducing appropriate theories such the parallel-DEVS [3] DS-DEVS [4], Cell-DEVS [5], etc.

Although these variants exist, it is very difficult to simulate complex systems, especially those represented by large-scale cellular-DEVS models. The need for computational resources becomes vital in order to ensure reliable and fast simulations. Many techniques were adopted with their advantages and disadvantages [6].

In this work, we take advantage of the property, closure under coupling of DEVS which allows us to reduce computational resources without changing the bases of DEVS specification.

A case study of the Wildfire Spread Simulation is modeled by the cellular-DEVS coupled model and its equivalent DEVS atomic model. Simulation results for both techniques are given and a comparison between these two simulations is described.

The remainder of this paper is outlined as follows: Section 2 resumes the basic concepts on the DEVS formalism. Section 3 is divided into two subsections, in the first one, a literature review on optimization methods to improve the performances of simulation within the DEVS formalism is reported, and in the second subsection, an overview on forest fire spread simulations is presented. Section 4 depicts our approach by converting the cellular-DEVS into its equivalent DEVS atomic model using the property closure under coupling applied to wildfire spread. Section 5 illustrates simulation experiments on our case study on forest fire spread, we specify both models: cellular-DEVS coupled model versus DEVS atomic model. At the end, conclusion and perspectives are given.

2 DEVS Formalism

2.1 Background

Being based on system theory, the DEVS formalism is one of the means theoretically well-grounded to express discrete event systems in a hierarchical and modular manner. A discrete event system is a dynamic system whose behavior is directed by the appearance of discrete events [7].

The use of discrete event-based simulation, rather than time-driven simulation has been proven to reduce computation time in many applications. DEVS formalism can be used to specify systems whose input, state and output trajectories are piecewise constant [8]. In addition, DEVS can give highest performance for simulation of continuous systems typified by spatial and temporal heterogeneity [9][10].

Theoretically, each system is characterized by two features: functional (behavioral) and structural aspects. At the lowest level, a basic part called DEVS atomic, describes the autonomous behavior of a discrete event system. At the highest level, DEVS coupled describes a system as modular and hierarchical structure [1][2].

A DEVS atomic model is based on continuous time, inputs, outputs, states and functions (output, internal and external transitions, life states). Formally, a DEVS atomic model is described by the following seven-tuple:

\[ M_{\text{devs}} = (X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, t_a) \]

Where \( X, S \) and \( Y \) are the sets of input events, states, and output events, respectively, \( \delta_{\text{int}} : S \rightarrow S \) the internal transition function, describes the state changes, that occurs when the elapsed time reaches the lifetime of the state, \( \delta_{\text{ext}} : Q \times S \rightarrow S \) is the external transition function, where
\[ Q = \{(s, e) | s \in S, e \in R^+, 0 \leq e \leq t_a(s)\} \]
is the total state set and \( e \) describes the elapsed time since the last transition of the current state \( s \), \( \lambda : S \rightarrow Y \): when the elapsed time reaches the lifetime of the state, this function generates an output event, and \( t_a : S \rightarrow R^+ \cup \infty \): time advance function, which is the lifespan of a state.

The DEVS coupled model allows formalizing the modeled system in a set of inter-connected and reused components. A DEVS coupled model is defined as an eight-tuple:

\[ CM_{devs} = (X_{self}, Y_{self}, D, \{M_d | d \in D\}, EIC, EOC, IC, Select) \]

Where \( X_{self} \): set of inputs events, \( Y_{self} \): set of outputs events, \( D \) : is the name set of sub-components, \( \{M_d | d \in D\} \): set of sub-components of DEVS models, \( EIC \): set of External Input Coupling, \( EOC \): set of External Output Coupling, \( IC \): defines the Internal Coupling, and \( Select : 2^D \rightarrow D \): defines a priority between simultaneous events.

### 2.2 Cellular-DEVS

Several DEVS formalisms have evolved compared to the classic DEVS to fit specific needs. Cellular-DEVS models have been created to model and simulate several phenomena such as fire propagation, traffic control, etc. [9].

Cellular-DEVS originates from the cellular automata formalism. The latter is based on discrete-time simulation which consumes the computation power to update all cells at each time-step. A decrease in the time-step used in the models would increase accuracy, but would also result in a longer simulation time and requires large computational resources. In fact, in many cases, there are few cells that are concerned to be updated at each time-step, which makes this manner inefficient.

In order to overcome this problem, cellular-DEVS was adopted to offer computational resources to the active cells that really execute state transitions and therefore avoid useless computation on inactive cells.

The cellular-DEVS formalism divides the spatial space into a set of identical cells where computations are done locally. A cell is considered as a DEVS atomic model which executes the local computations based on its own state as well as its neighbor states. The spatial space is implemented as a DEVS coupled model where the internal couplings between cells are given by neighborhood rules [6].

### 2.3 Closure under Coupling

The property closure under coupling, in DEVS and parallel-DEVS, reports that every coupled model has its own equivalent atomic model. Therefore a DEVS coupled model regroups several DEVS models, which can be regarded as another DEVS atomic model.

The transition from the parallel-DEVS coupled model into its equivalent parallel-DEVS atomic model is described as follows [11, 6]:

The state set \( S \) of the subsequent model will be the Cartesian product of the total state sets of all the DEVS atomic models. Thus, the time advance \( t_a(s) \) defines the time remaining to the next event in component \( d \).

\[
S = \times_{d \in D} Q_d \\
t_a(s) = \text{minimum}\{\rho_d | d \in D\} \\
\text{Where} \\
s \in S, s = (\ldots, (s_d, e_d), \ldots) \text{ for all } d \in D \text{ and } \rho_d = t_a(s_d) - e_d
\]

The overall transition function is defined as:
\( \delta(s, e, x^b) = \begin{cases} 
\delta_{\text{ext}}(s, e, x^b) & \text{if } 0 \leq e < t_a(s) \text{ and } x^b \neq \phi \\
\delta_{\text{conf}}(s, e, x^b) & \text{if } e = t_a(s) \text{ and } x^b \neq \phi \\
\delta_{\text{int}}(s) & \text{if } e = t_a(s) \text{ and } x^b = \phi 
\end{cases} \)

Where

\( \delta_{\text{ext}} : \times_{d \in D} Q_d \times X \rightarrow \times_{d \in D} Q_d \)

\( \delta_{\text{conf}} : \times_{d \in D} Q_d \times X \rightarrow \times_{d \in D} Q_d \)

\( \delta_{\text{int}} : \times_{d \in D} Q_d \rightarrow \times_{d \in D} Q_d \)

\( \lambda : \times_{d \in D} Q_d \rightarrow Y \)

DEVS coupled model can be reduced to a behaviorally equivalent DEVS atomic one. Thus, the concept of closure under coupling ensures that the coupled model results in a model of the same class which has a basic specification.

3 Related Works

3.1 Optimization Techniques in DEVS Simulation

In many situations, simulate complex systems with DEVS requires machines with very high performance. To overcome this crucial need, various techniques have been used and can be classified in two categories: those that modify the DEVS formalism by adding either specific functions or variables to manage in a dynamic way the structure of the models, which require demonstrating again some of the properties of DEVS (closure under coupling, hierarchy, etc.), and those that preserve the DEVS formalism but which integrate information concerning the behavioral model structure, which decrease the modularity, flexibility and its reuse.

For both categories, different techniques have been developed in the DEVS community such as quantization [12], which helps in enhancing performance simulation by decreasing the number of state transitions and messages. However, [6] have noted that quantization is an approximation method that satisfies the tolerance requirement, i.e, the simulation should stay within acceptable error.

The parallel and distributed DEVS simulations need more hardware as well as extra work for parallelization of prevailing sequential models [13, 14, 7]. However, [15] have noted that, the overhead of check pointing and rollback operations may result in unstable and degraded performance.

DEVS components within variable structure [16], permit models and couplings to be dynamically added and/or removed during simulations run. This main advantage is loosen when an important number of update (add and/or remove) of models and couplings are done during a simulation cycle [17].

However, there are few works based on converting DEVS coupled into DEVS atomic to increase performance [6]. Among these works, [7] presented a formal approach of this property with addition of scheduling mechanism, they introduced a composition-based method that converts a DEVS coupled model into its equivalent DEVS atomic model at compile time in order to speed up the simulation by accounting events and messages at compile time. The main advantage is that the resultant atomic model should keep track of all the functions of these internal models. However, this technique results in overhead as the number of the internal models grows up, and the conversion process will be more complex to be done properly. [18] followed the same approach by converting a coupled model into atomic one in order to eliminate the message overhead based on
Modelica parallel variable update. Unfortunately, this approach shows slow simulations because of the large number of variables used. Thereby, these two techniques proceed at compile time to allow the conversion.

In [6], the idea is to convert a set of cellular-DEVS atomic model into its single DEVS atomic model while keeping the same accuracy and remaining an error-free approach. Thus, instead of considering each cell as an atomic model, with the closure under coupling and at the specification and development stage, the method will transform a group of cells into a non-hierarchal and non-modular model which leads to increase the performance simulations run.

Although the resulting model is not hierarchical, but this does not present any problem to the user since they are identical sub-models. Therefore, the user will bypass the hierarchy due to the iterative and repetitive behavior and structure of cells.

The present work is closely related to [6]. Our work focuses mainly on the usage of the memory resource. We apply the property closure under coupling in the modeling of forest fire to obtain reliable simulations by decreasing the use of memory resource; and as a collateral profit, we speed up the simulation run and we increase considerably the number of concurrent simulated cells.

### 3.2 Forest fire spread

Wildfires continue to cause considerable losses of human lives, wildlife and houses every year and are costly to contain. In United States of America alone, about one billion dollars is spent annually on wildfire suppression and containment [19, 20].

Fire spread is a propagation process and needs to build simulation models that relate the system evolution as accurately as possible to plan scenarios in order to save lives and money. However, simulating wildfire spread remains a challenging problem due to the complexity of wildfire behavior.

Such far, no model has tried to combine the different features of fire behavior (vegetation, weather, topography, etc.) that are already established separately because of the important factors that influence wildfire behavior and the interaction between them [21, 19], that why there are many models in the literature.

The most common approaches to simulate fire spread are based either on vector or wave approach, or on the cellular models [21]. For the former, the model produces vector fire perimeters at determined time intervals. For the latter, the cellular models simulate fire spread as a discrete process of ignitions across a regular division of the space in cells. Successive computations are carried out on each cell to ignite its neighbors.

The majority of fire models in use today are principally based on fire propagation relations developed by [22], [23] and [24]. The popular Rothermel model was in particular chosen because of its robustness and stability which have been extensively tested and proven [25, 19].

Both categories of models (vector, cellular) are simulated either in discrete or continuous time [26]. The following non-exhaustive list of examples uses the Rothermel model. HFIRE [27] includes discrete time model, FARSITE [21] where the fire growth is based on Huygens principle of wave propagation, and BEHAVE [28] uses continuous simulation for the vector models.

The forest fire simulation models include cellular space, cellular automata [25], cellular-DEVS models [29], DEVS-Fire [30], and cell-DEVS which uses heat transfer partial differential equations to calculate fire spread in each cell [31, 32].

Cellular-DEVS and Cell-DEVS simulation results were compared and the analysis concluded the validation of both models [32]. However, the present cellular implementation knows a simulation issue whenever all cells in the cell space are created simultaneously which causes
simulation performance degradation. This degradation is attributed to the initialization of the cells and the functions needed, and the memory that is required to run the simulation [19].

A dynamic structure was proposed to overcome this issue [19, 20]. This approach keeps track of active cells along the fire front as the simulation proceeds. The cells are created/deleted dynamically; however this implementation has computational overhead at runtime when we proceed with a large-scale cell space.

Other implementations were proposed to overcome these concerns. Indeed, besides using high performance computers, researchers still develop other techniques including innovative algorithms that allow handling only active cells or research on how the non-modularity/non-hierarchy in modeling can improve simulations run [6, 33].

4 Formal Specification and Design Issue

The M&S of forest fire spread is confronted to size of the map in a geographic information system (GIS). So, obtaining reliable and accurate simulations needs huge computation performance. The aim of this section is to show the benefit of applying the closure under coupling property of DEVS to enhance the use of memory space and increase the size of the map on GIS during simulation runs.

4.1 Problem Description

The forest is represented as a 2D cell space of square cells whose dimensions depend on the resolution of the GIS. Each cell has eight neighbors and carried out its local computation of the rate of fire spread and direction based on its local conditions (parameters). The literature classifies the parameters which set the fire spread ratio into three groups: vegetation type (caloric content, density, etc.), fuel properties (vegetation size) and environmental parameters (wind speed, humidity and slope). The flaming fire evolves principally according to the wind speed and its direction [34].

We assume, in this work, that fuel, topography and weather conditions are uniform for each cell. This work is closely related to [19, 20]. The rate of fire spread of each cell relies on Rothermel model [22] which is decomposed thereafter into eight spreading directions according to the cell neighborhood and schedules ignition of its neighboring cells accordingly.

The ignition process consists on spreading fire from a burning cell to its neighbors using Moore neighborhood. A cell space is considered as a DEVS atomic model, therefore, the forest cell space is a coupled model composed of a set of cell atomic models.

We consider that each cell can be in one of the following possible states:

- **Nonflammable (N)**: It can be a road, a surface of water or just an empty surface.
- **Unburned (U)**: Passive state; it represents any fuel which is not consumed yet by fire.
- **Burning (B)**: represents a consuming fire.
- **Ash (A)**: It is afterburning state, it is the final combustion process state.

4.2 Functional Architecture

The overall system proposed in this work is composed of (Figure 1):
• Forest cell space, in which each forest cell is an atomic model.

• Geographic information system interface which provides the different data that influence the spread rate. It consists of:
  – Environmental parameters interface: provides weather and topography data such as wind speed, its direction, relative humidity, slope, aspect, etc.,
  – Fuel properties: provide size of vegetation, fuel model, etc., and
  – Vegetation type: provides density of vegetation, its calorific content, etc.

• Forest cell igniter: ignites an initial set of cells to start the simulation, and

• Rothermel model: computes fire spread and its direction.

In its basic form, Forest cell space is modeled as a grid composed of \( n \) rows and \( m \) columns, which depends on the data resolution. The dynamic system of the flaming front propagation speed is given by the simulator. It is based on the current cell position and its own variables, each of which is given by an appropriate model. The forest cell igniter is a DEVS atomic model; it is coupled to all forest cells to ignite cells. The Environmental interface and Fuel and Vegetation interface provide two kinds of values: spatial-temporal and environmental data. These data are fed into the forest cells which in turn send these data to Rothermel model to compute the rate of fire spread and its main direction.

### 4.3 Formal Specification and Implementation

In this section, we are going to describe the overall system design. As seen above, the fire spread model is composed of forest cell space coupled to the igniter model, Rothermel model, and GIS...
interface (environmental, vegetation and fuel models).

4.3.1 Forest cell DEVS atomic model

For each cell \( O = (a, b) \), its Moore neighborhood is given by the set:
\[
V = \{O, N, NE, E, SE, S, SW, W, NW\}
\]

Each neighbor is reported as one of eight compass points \((N, NE, E, SE, S, SW, W, NW)\). In Figure 2, the fire is propagated from the center of the cell to the center of the neighbor cells.

![Figure 2: Cell center-to-center fire spread.](image)

Each forest cell DEVS atomic model has four states: Nonflammable (N), Unburned (U), Burning (B) and Ash (A). All cells are assumed to be initially in unburned state, except those defined by the user in the passive state Nonflammable.

Once a cell is ignited it computes the ignition delays for its eight neighbors, and sends out the ignition message for each of them. Therefore, each cell is ignited either by its neighbor or by the igniter model. The ignited cell, in its turn, ignites its neighbors. Thus, an additional input port is coupled with each forest cell DEVS atomic model. The ignition delays are calculated by the Rothermel model. Each cell sends out its state variables (fuel model, size of vegetation, wind speed, wind direction, etc.) obtained from the GIS interface toward the Rothermel model. The latter sends out as a result the rate of fire spread and its main direction to the concerned cell. Therefore the forest cell interacts with the weather and fuel interface, and the Rothermel model via these additional input/output ports.

Consequently, each forest cell has 11 input ports and 9 output ports by which it reacts and acts on its environment (Figure 3).

The dynamics of fire spread is described as follows: once a cell is ignited, it sends out ignition messages one by one in each of the eight directions. The decomposition of fire spread in each direction is based on the model that defines the fire shape as an ellipse \([20, 21]\). The contagion process of fire spread across a cell considered in our case study is center-to-center as in [26]. Consequently, the burning cell remains in this state at least until all the eight messages will be sent out.

Let us consider:

* \( T_{td} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\} \), the set of these eight different time delays sorted respectively from the smallest to the largest.
Figure 3: Forest cell DEVS atomic model.

- \( \Delta T_{td} = \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, \rho_7, \rho_8\} \),
  the set of the time interval between two adjacent ignition messages.
  \( \rho_1 = t_1 \) and \( \rho_i = (t_i - t_{i-1}) | i = 2 \ldots 8 \).

The first message is sent at \( t + t_1 \) and the last one at \( t + t_8 \), at this moment, the forest cell transitions from the burning state to the ash one (burned) after sending out its last ignition message. The set \( T_{td} \) can be affected by the parameters change (wind flow, speed, direction) and consequently the set \( \Delta T_{td} \). In this case, the forest cell space model notifies the Rothermel model which updates the rate of fire spread and subsequently the spread in each direction. The rate of fire spread in each direction is depicted on Figure 4.

Figure 4: Rate of spread calculation in eight directions.
The formal specification of the forest cell DEVS atomic model (Figure 5) is:

\[ ForestCellAM = (X, S, Y, \delta_{\text{ext}}, \delta_{\text{int}}, \delta_{\text{conf}}, \lambda, t_a) \]

where:

- \( X \) in Ports \( = \{ \text{InN}, \text{InNE}, \text{InE}, \text{InSE}, \text{InS}, \text{InSW}, \text{InW}, \text{InNW}, \text{InIgn}, \text{InPar}, \text{InRot} \} \)
- \( X = \mathbb{R}^9 \times \mathbb{R}^d \times \mathbb{R}^2 \)
- \( Y \) out Ports \( = \{ \text{OutN}, \text{OutNE}, \text{OutE}, \text{OutSE}, \text{OutS}, \text{OutSW}, \text{OutW}, \text{OutNW}, \text{OutPar} \} \)
- \( Y = \mathbb{R}^8 \times \mathbb{R}^d \)

\( d \) is a dimension, that represents the cell parameters, its output is coupled with Rothermel model.

\[ X_{\text{in Ports}} = \mathbb{R}^9 \times \mathbb{R}^d \times \mathbb{R}^2 \]

Where dimension 9 represents the nine ignition messages (8 from neighbors and one from igniter model), \( d \) is the number of parameters of each forest cell (weather, wind, humidity, etc.), and the number 2 is the couple, rate of fire spread and its main direction.

\[ \delta_{\text{ext}}(\text{Unburned}, \rho, (\text{InN}?)x) = (\text{Burning}, \rho_1) \]

\[ \delta_{\text{ext}}(\text{Burning}, \rho, (\text{InPar}?)x) = (\text{Burning}, \rho_i) \]

\[ \delta_{\text{conf}}(s, \rho, x) = \delta_{\text{ext}}(\delta_{\text{int}}(s), 0, x) \]

\[ \lambda(\text{Burning}) = \text{Out}!\text{ignition} \]

The initial state of this model is \((\text{Unburned}, \infty)\).

Where \( \text{In} \in X_{\text{in Ports}} - \{ \text{InPar}, \text{InRot} \}, \rho_i \) is computed by the Rothermel model whenever parameters change, \( \text{Out} \in Y_{\text{out Ports}} - \{ \text{OutPar} \}, \text{ignition} \in R \).
In Figure 5, the solid line describes the occurrence of external event, while the dashed line depicts the internal state changes, and occurs when the elapsed time reaches the lifetime of the state.

### 4.3.2 Forest Fire Spread and Decomposition

The Rothermel mathematical model [22] computes a one-dimensional maximum fire spread rate and its direction. Fire geometry has been determined empirically [21]. However, in the literature, the fire shape is considered as an ellipse which is widely used [24]. Therefore the fire spread is inferred in all directions from the 1-D maximum rate using the mathematical properties of the ellipse and the equations defined in [21, 25]. The spread rate in an arbitrary direction $\theta$ (Figure 6) is obtained by:

$$
R(\theta) = \frac{R_{\text{max}}}{1 - \epsilon \cos \theta}
$$

$$
\epsilon = \frac{\sqrt{lw^2 - 1}}{lw}
$$

$$
lw = 0.936e^{0.2566v} + 0.461e^{-0.1548v} - 0.397
$$

Where $R_{\text{max}}$ is maximum rate of spread, $lw$ is the ellipse ratio of the semi-major over semi-minor (Length to breath ratio) and $v$ is the midflame wind speed (effective wind speed).

Once the rate of spread is known in all directions, we compute the time delays to reach the center of the neighbors by this equation:

$$
t_i = \frac{d_i}{R_i} | i = 1..8
$$

Where $t_i$ is the time to reach the neighbor $i$, $d_i$ is the distance from the center of the burning cell to the center of its neighbor $i$ and $R_i$ is the fire spread rate in this direction. In case of parameters change (wind speed, direction, humidity, etc.), the new $R_{i,\text{new}}$ in direction of the neighbor $i$, is inferred from Rothermel model if the fire hasn’t yet reached it. The new remaining time delay is:

$$
t_{i,\text{new}} = \frac{\Delta d_i}{R_{i,\text{new}}}
$$

Figure 6: Rate of spread according to an arbitrary direction.
Where \( \Delta d_i \) is the remaining distance to reach the neighbor \( i \) and \( t_{i_{\text{new}}} \) is the new time delay. As the fire propagates, the equation allows the model to be dynamic and sensitive to weather changes.

### 4.3.3 System Formal Design by Closure under Coupling

In the forest cellular approach, the forest fire spread is usually presented as a set of arranged cells whose dimensions depend on the resolution of GIS [20]. Thus, forest cell space coupled model is a discrete dynamical system formed by coupling a finite number of forest cells. These cells are arranged uniformly in a two-dimensional space composed of \( n \) rows and \( m \) columns. Each forest cell atomic model is coupled to 8 neighbors as described in Figure 7.

![Forest cell space DEVS coupled model](image)

**Figure 7:** Forest cell space DEVS coupled model.

The fire spread model is described in Section 4.2. It is composed of forest cell space (a grid of forest cells), igniter model, GIS interface and Rothermel model. However, simulate such a model will be affected by large-scale forest cell model and particularly during the simulation run and involves more memory usage. A huge memory is needed to simulate the overall forest and if you lack memory, you simulate just a part of it. In [19], memory required for some implementations is illustrated.
To overcome to this issue, we propose to use the closure under coupling. This property of the DEVS formalism consists to transform a coupled model into its equivalent atomic model. Thus, the forest cell space coupled model will be converted into an atomic model.

Each cell becomes a state variable of this atomic model and accedes to its neighbor state directly by removing the ports from the atomic models that is resulted by small volume of inter-component messages during simulation run. In fact, the communication between cells is done directly inside the model, except outdoor events [6].

Based on the Forest Cell DEVS atomic model described in Section 4.3.1, the forest cell space DEVS coupled model is transformed into forest cell space DEVS atomic model and would benefit from the property of closure under coupling. It is described as follows:

\[
\text{ForestCellSpaceAM} = (X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \delta_{\text{conf}}, \lambda, t_a)
\]

- \(X = \{(\text{ignite}, \text{list}), (\text{inParam}, \text{parameters})\}\) — list is a list of initial ignited cells
- \(Y = \emptyset\), there is no output event to send out.
- \(S = \text{Cell} \times \text{inParam} = \{\ldots, (\text{phase}_{i,j}, \rho_{i,j}), \ldots\} \times \mathbb{R}^d| i = 1 .. n, j = 1 .. m\)

Where \text{Cell} is the set of forest cell. Each cell is identified by its position (row, column) which can be in one of the four states (Nonflammable, Unburned, Burning, Ash) and \(d\) is the number of parameters provided by GIS interface.

Each cell keeps its current state and according to external and internal events state changes will occur to update the state of the concerning cell. It is pointed out that each cell has eight neighbors except those situated on the bound of the forest. Environmental parameters are assumed to be uniform at each cell except the wind which can change its speed and direction over time. Thus, the fire spread direction can change dynamically. Thus, we get a non-modular modeling structure (see Figure 8). With a modular modeling, when a cell ignites its neighbors, it invokes internal state change to provoke the execution of the output function \(\lambda()\), which sends out the event ignition to all neighbors. On the other hand, the non-modular modeling propagates the output event ignition directly to neighboring cells without calling the simulator for dispatching events. Such a communication optimizes the DEVS M&S structure and decreases the number of exchanged messages between cells.

Consequently, the fire spread model in its non-modular structure is comprised of four atomic models: Igniter atomic model, Forest cell space atomic model, GIS interface and Rothermel model as shown on Figure 9. It is identical to the structure depicted in Figure 1 except the Forest Cell Space which is converted to an atomic model.

Note that this architecture remains an open and modular one which allows enhancing the functionalities of the GIS and updating the Rothermel package.

5 Simulation Experiments

In this section some output results that measure the simulation performance of the two models are provided. The package Behave was used to compute the rate of fire spread given by the Rothermel model. The experiments were carried out on laptop Dell T2300@1.66GHz with Intel CPU, 1.66 GHz processor, 2.50 GB of RAM and Windows XP© operating system and as a simulator, we have used our own simulator developed in Java and respecting DEVS specifications.
\(\delta_{\text{ext}}(s, e, x)\)  
\(c, c' : \text{Cell}\)  
if \((x = \text{ignite})\)  
\{  
for each \(c \in \text{ignite}\)  
\(s_c = \delta_{\text{ext}}(s_c, e, \text{ignite})\)  
\}  
if \((x = \text{inParam})\)  
update parameters  
recompute the lifetime for each active cell \(c\)  
\(\delta_{\text{int}}(s)\)  
for each \(c \in \text{Cell}\)  
\{  
if \((\text{lifetime}(c) = \text{lifetime}(\text{Cell}))\)  
\{  
if \((\text{burning}(s_c))\)  
\{  
for each \(c' \in \text{neighbor}(c)\)  
\(s_{c'} = \delta_{\text{ext}}(s_{c'}, \text{lifetime}(\text{Cell}), \text{ignite})\)  
\}  
\}  
else  
\(\text{lifetime}(c) = \text{lifetime}(c) - \text{lifetime}(\text{Cell})\)  
\}  
\(\delta_{\text{conf}}(s, e, x)\)  
\(\delta_{\text{ext}}(\delta_{\text{int}}(s), 0, x)\)  
\text{lifetime}(s)\)  
return \(\min\{\text{lifetime}(c) \mid c \in \text{Cell}\}\)

Figure 8: DEVS atomic model functions of forest fire spread

Figure 9: The forest fire spread model via closure under coupling.

5.1 Comparison Results

As mentioned by [19], an assessment of memory usage based on the implementation in [35] shows that each cell needs about 35 kB memory space. This fact is time consuming and resource intensive and therefore not very effective for large-scale cells simulation.

We were tempted to see the limits of our simulator for these two specifications of forest fire spread. The ForestCellSpace DEVS coupled model versus ForestCellSpace DEVS atomic model.

Table 1 summarizes some important results. The model ForestCellSpace coupled model uses a
Table 1: Comparison results.

| Results                                      | ForestCellSpace Coupled Model | ForestCellSpace Atomic Model |
|----------------------------------------------|-------------------------------|-----------------------------|
| Maximum number of simulated cells [cells]     | 70 × 70                       | 2682 × 2682                 |
| Used Memory [MegaBytes (MB)] for 60 × 60 cells | 8.74                          | 1.23                        |
| Used Memory [MB] for 600 × 600 cells         | Out of Memory Error           | 12                          |
| Simulation Execution Time [second (sec)] for 60 × 60 cells | 109,131                      | 13,390                      |

Table 2: Sample of simulation performances of ForestCellSpace: Coupled vs. Atomic

| Indicator                                      | ForestCellSpace Coupled Model | ForestCellSpace Atomic Model |
|------------------------------------------------|-------------------------------|-----------------------------|
| Simulation Execution Time [sec.]               | 109,131                       | 13,390                      |
| Used Heap Memory [MB]                          | [2, 20]                       | [2, 8]                      |
| Thread Count                                   | [9, 3500]                     | [9, 14]                     |
| CPU Usage [%]                                  | 100                           | 100                         |

conventional architecture, where a grid of cells is used. While ForestCellSpace atomic model uses the closure under coupling to get a unique atomic model of the forest cell space.

The experiment revealed that only about 70 × 70 cells of the overall forest space has been simulated in coupled model, whereas the other specification, i.e., forest cell atomic model, a limit of 2682 × 2682 cells was reached therefore, we get a ratio close of 1 over 39 for each cell.

Another advantage of the forest cell space atomic model is the simulation time; in fact, the latter is 8.15 times faster than the forest cell space coupled model.

In order to get more results, we have used additional free software Java Virtual Machine Monitor (JVM Monitor) to profile our Java application. The snapshot of the Figure 10(a) resumes some characteristics of simulation of ForestCellSpace coupled model; whereas Figure 10(b) gives the values of these characteristics of ForestCellSpace atomic model.

Through these Figures 10(a) & 10(b) and by looking inside execution time windows of each simulation, we can see the advantages of the usage of closure under coupling property. We have less amount of used heap memory. The maximum values reached with Forest Cell Space Coupled Model are greater in overall cases compared to Forest Cell Space Atomic Model.

Table 2 gives the performance indicators of both Forest Cell Space simulations coupled and atomic at some critical times which are extracted from curves of Figure 10 in case of 60 × 60 cells.

As mentioned before, despite the different implementations of forest fire spread, it is essential to remind that both models have the same behavior, whereas their simulation performances are different (see Figure 11 for a cell space of 60 × 60).

The front flame is depicted in color yellow while the black color is dedicated to the ash area. The colors pink, orange, red and light gray represent the different phases of the burning state (flanking and backing spread).
(a) ForestCellSpace coupled model: grid of cells.

(b) ForestCellSpace atomic model: closure under coupling.

Figure 10: Performances of forest fire spread simulation.
5.2 Verification and validation

The validation of fire spread model is not an easy task, and few works describe validations based on real fire [25]. Nevertheless, we are going to do a partial validation by comparing the results of our approach by those of FARSITE. The latter is the most generally distributed and accepted fire behavior predictive model used in forestry [19].

On one hand, we are going to simulate our model described in Section 4.2 by adopting this functional architecture, on the other hand, we are going to simulate by FARSITE software the same scenario and compare the different results. Therefore, the equivalence of the two forms is verified using simulation methods.

In these simulations, we suppose that:

- Virtual forest is constructed as a grid of $260 \times 260$ cells where each cell represents an area of $1 \times 1 \text{ m}^2$ (m: meter).
- Fire spreading is on each of the eight compass points.
- Starting ignited point is the cell(130,30).

We assume that uniform parameters characterize the cell space, i.e., each cell has uniform fuel, terrain, weather and fire behavior along the forest fire area.

The fire shape is assumed elliptical and the decomposition scheme considered is center-to-center. We use the Rothermel model to compute the maximum fire spread rate. Fire spread in all other directions is inferred from the spread rate using the mathematical properties of the ellipse (equation[1]).

We consider the following values to get the rate of spread by the Rothermel model:

- The model of fuel considered in this experiment is Northern Fire Forest Laboratory (NFFL) Fuel Model 1: short grass (1 feet (ft)) with its standard characteristics.

Besides that, we assume:
- wind speed [meter/second (m/s)] $wsp = 1.0$
- wind direction [degree(°)] $wdr = 0.0$
- slope [°] $slp = 16.7$
- aspect [°] $asp = 0.0$

The Rothermel model produces the following results:
- rate of spread [m/s] $ros = 0.26$
- direction of maximum spread [°] $sdr = 180$
- effective wind speed [m/s] $efw = 1.37$
Table 3: Spreading rate in the eight directions

| $\theta$ | $R(\theta)$ | % $T_{OS}$ |
|----------|-------------|------------|
| 0        | 0.26        | 100        |
| ±45      | 0.14        | 53.77      |
| ±90      | 0.066       | 25.41      |
| ±135     | 0.04        | 16.63      |
| 180      | 0.038       | 14.55      |

Figure 12: Fire spread progression (260×260 cells)

Length to breath ratio $lw$ is obtained via equation (3), its value for these parameters is:

$- lw = 1.50$

The spreading rate in each direction is given by equation (1) based on the ratio $lw$. Thus, we obtain the following spreading rates according to each direction as shown in Table 3.

According to these results, we simulate our forest fire model. Figure 12 shows the fire spread taken at different simulation times and gives us the effective burned area [hectare (ha)].

On the other hand, we have done the simulation on FARSITE with the same parameters and we obtained output results shown on Figure 13.

Figure 13: Fire spread progression on FARSITE

A comparison was done on the progression of fire area over time (see Figure 14). The result shows the likeness between both methods. Of course, the goal is not to mimic FARSITE software behavior, it is possible to fine-tune or parameterize our model so that it mimics FARSITE and
obtain the same graph. However, the real purpose is to show the vitality of our approach. As you can see, the same scenario can’t be simulated with the forest cell space coupled model and we will get a message as Out of memory error. A solution suggested is, either we divide this forest space into small areas or decrease the cell resolution (increase cell dimension) which may result in incorrect results and degrade accuracy.

As cited previously, this is a partial and preliminary validation. More validation is definitely required because of the many potential sources of error which can confuse the comparisons.

These tests have underlined the necessity to consider errors associated with:

- The spatial and temporal resolution of the inputs,
- The model input data (fuels and environmental parameters) and
- The nature of fire growth projections used for comparison in both methods (Huygens principle of wave propagation in FARSITE versus Rothermel spread equation in Behave package).

Logically, a wildfire spread simulation should be most accurate when using accurate data at high spatial and temporal resolutions.

The acceptability of this statement is dependent on the spatial resolution required by the user and the resolution specified for the simulation. Thus, the sensitivity of fire simulation to the resolution, fire spread patterns generated (Huygens principle) and model of different fire input parameters remain to be verified.

However, it should be noted that even FARSITE needs to be validated [21], the most important result of FARSITE tests to date has been proven that spread rates for all fuel models tended to be over predicted by the Rothermel spread equation. Different adjustments have been added in order to correct the observed error with real forest fire, but there is no way to guide settings for these adjustments. Consequently, future work needs to focus more on validating our approach using observed fire growth patterns in several conditions of weather and vegetation.

6 Conclusion and Future Works

This work attempts to foster the integration between two concepts: closure under coupling in cellular-DEVS and the forest fire spread to gain memory resource during simulation run.
The study of the simulation of forest fire has increased considerably, it requires building simulation models that allows for system evolution in both time and space.

The most common approaches to simulate fire spread are based either on vector approach, or on the cellular models. Cellular models simulate fire spread as a discrete process of ignitions across a regular division of the space in cells. However, dealing with high cell resolutions naturally defies efficient computer simulation.

In this work, we tried to gain some performance in simulation run. This was achieved by converting the cell space into atomic model in order to eliminate inter-cell messages. This approach was initially deduced from the property closure under coupling of the DEVS formalism. Some relevant results are presented and a comparison between our approach and the simulation by FARISTE software is given.

However, this implementation needs an additional work in verification and validation (which we crossed an important step) to confirm more again the vitality of this approach. The functional architecture needs to take into account more parameters since the fire spread process might become complicated. A model for these parameters must be coupled to our architecture in order to get more realistic results. Also we need to validate our approach by using observed fire growth patterns in several conditions of weather and vegetation. An issue that we hope explore in the near future.

References

[1] B. P. Zeigler, Theory of Modeling and Simulation, 1st ed., Wiley InterScience, New York, 1976.
[2] B. P. Zeigler, H. Praehofer, T. G. Kim, Theory of Modeling and Simulation, 2nd ed., Academic Press, 2000.
[3] A. C. Chow, Parallel-DEVS: a parallel, hierarchical, modular modeling formalism and its distributed simulator, Transactions of the Society for Computer Simulation International 13 (1996) 55–67.
[4] F. J. Barros, Modelling formalisms for dynamic structure system, ACM Transactions on Modelling and Computer Simulation 7 (1997) 501–515.
[5] G. Wainer, N. Giambiasi, Application of the cell-DEVS paradigm for cell spaces modelling and simulation, Simulation: Transactions of the Society for Computer Simulation International 71 (2001) 22–39.
[6] F. A. Shiginah, B. P. Zeigler, A new cell space DEVS specification: Reviewing the parallel-DEVS formalism seeking fast cell space simulations, Simulation Modelling Practice and Theory 19 (2011) 1267–1279.
[7] W. B. Lee, T. G. Kim, Simulation speedup for DEVS models by composition-based compilation, in: Summer Computer Simulation 2003 (SummerSim2003), 2003, pp. 395–400.
[8] B. P. Zeigler, H. S. Song, T. G. Kim, H. Praehofer, DEVS framework for modeling, simulation, analysis, and design of hybrid systems, Lecture Notes in Computer Science, Springer-Verlag 999 (1995) 529–551.
[9] X. Hu, B. P. Zeigler, A high performance simulation engine for large-scale cellular DEVS models, in: High Performance Computing Symposium (HPC’04), Advanced Simulation Technologies Conference, 2004.

[10] J. Nutaro, B. P. Zeigler, R. Jammalamadaka, S. Akerkar, Discrete event solution of gas dynamics within the DEVS framework: Exploiting spatio-temporal heterogeneity, in: International Conference on Computational Science (ICCS’03), Melbourne, Australia, 2003.

[11] F. A. Shiginah, Multi-Layer Cellular DEVS Formalism for Faster Model Development and Simulation Efficiency, Ph.D. thesis, Department of Electrical and Computer Engineering, University of Arizona, USA, 2006.

[12] E. Kofman, S. Junco, Quantized-state systems: A DEVS approach for continuous system simulation, Simulation: Transactions of the Society for Computer Simulation International 18 (2001) 123–132.

[13] Y. H. Wang, B. P. Zeigler, Extending the DEVS formalism for massively parallel simulation, Discrete Event Dynamic Systems: Theory and Applications 3 (1992) 193–218.

[14] S. Park, Hierarchical Model Partitioning for Distributed Simulation of Hierarchical and Modular DEVS Models, Ph.D. thesis, University of Arizona, USA, 2003.

[15] G. Wainer, K. Al-Zoubi, O. Dalle, D. Hill, S. Mittal, J. Martin, H. Sarjoughian, L. Touraille, M. Traore, B. Zeigler, Standardizing DEVS Model Representation, CRC Press, USA, 2010.

[16] X. Hu, B. P. Zeigler, S. Mittal, Variable structure in DEVS component-based modeling and simulation, Simulation: Transactions of the Society for Computer Simulation International 8 (2005) 91–102.

[17] Y. Sun, X. Hu, Performance measurement of dynamic structure DEVS for large-scale cellular space models, Simulation: Transactions of the Society for Modeling and Simulation International 85 (2009) 335–351.

[18] T. Beltrame, Design and Development of a Dymola/Modelica Library for Discrete Event-oriented Systems Using DEVS Methodology, Master thesis, Department of Computational Science, ETH Zurich, USA, 2006.

[19] L. Ntaimo, X. Hu, Y. Sun, DEVS-FIRE: Towards an integrated simulation environment for surface wildfire spread and containment, Simulation 84 (2008) 137–155.

[20] X. Hu, Y. Sun, L. Ntaimo, DEVS-FIRE: design and application of formal discrete event wildfire spread and suppression models, Simulation: Transactions of the Society for Modeling and Simulation International 88 (2012) 259–279.

[21] M. A. Finney, FARSITE: Fire Area Simulator-Model Development and Evaluation, Technical Report RMRS-RP-4 Revised, USDA, 2004.

[22] R. C. Rothermel, A mathematical model for predicting fire spread in wildland fuels, Technical Report 115, Ogden, UT, U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station, Arizona, USA, 1972.

[23] F. Albini, Estimating wildfire behavior and effects, Technical Report INT-30, USDA For. Serv., 1976.
[24] H. Anderson, Predicting wind-driven wildland fire size and shape, Technical Report INT-305, USDA For. Serv., 1983.

[25] D. D’Ambrosio, S. D. Gregorio, W. Spataro, G. Trunfio, A model for the simulation of forest fire dynamics using cellular automata, in: Third Biennial Meeting: Summit on Environmental Modelling and Software Society (SEMSS2006), Burlington, USA, 2006.

[26] L. Ntaimo, B. Khargharia, Two-dimensional fire spread decomposition in cellular DEVS models, in: Spring Simulation Multiconference (SpringSim2006), Society for Computer Simulation International, Huntsville USA, 2006, pp. 103–109.

[27] M. Morais, Comparing Spatially Explicit Models of Fire Spread through Chaparral Fuels: A New Model Based Upon the Rothermel Fire Spread Equation, Master thesis, University of California, Santa Barbara, USA, 2001.

[28] R. E. Burgan, R. C. Rothermel, BEHAVE: Fire Behavior Prediction and Fuel Modeling System– FUEL Subsystem, Technical Report PMS 439-1, National Wildfire Coordinating Group, USA, 1984.

[29] J. M. Vasconcelos, B. Zeigler, J. Pereira, Simulation of fire growth in GIS using discrete event hierarchical modular models, Advances in Remote Sensing 4 (1995) 54–62.

[30] Forest fire spread and suppression in DEVS, Simulation: Transactions of the Society for Modeling and Simulation International 80 (2004) 479–500.

[31] J. Ameghino, A. Troccoli, G. Wainer, Models of complex physical systems using cell-DEVs, in: Proceedings of Annual Simulation Symposium, 2001, 2001.

[32] A. Muzy, G. Wainer, E. Innocenti, A. Aiello, J. Santucci, Comparing simulation methods for fire spreading across a fuel bed, in: Proceedings of International Conference on Autonomous and Intelligent Systems (AIS2002), 2002, pp. 219–224.

[33] M. Hamri, Y. Dahmani, A non-modular modeling and simulation approach based on DEVS for the forest fire spread, in: Fifth International Conference on Advances in System Simulation, 2013, IARIA, Venice, Italy, 2013, pp. 130–136.

[34] J. Scot, R. Burgan, Standard fire behavior fuel models: a comprehensive set for use with Rothermel’s surface fire spread model, Technical Report RMRS-GTR-153, USDA Forest Service General, USA, 2005.

[35] DEVSJAVA, DEVS M&S tool, 2009. Retrieved March 1, 2013 from http://www.acims.arizona.edu/software/software.htm.