Expressing a Forest Cell Model in Parallel DEVS and Timed Cell-DEVS Formalisms

Lewis Ntaimo and Bernard P. Zeigler

Arizona Center for Integrative Modeling and Simulation
Department of Electrical and Computer Engineering
The University of Arizona
1230 E. Speedway, Tucson, AZ 85721
ntaimo@email.arizona.edu

Keywords: Forest cell, fire spread, Parallel DEVS, Timed Cell-DEVS, model correctness

Abstract

One ecological problem that has raised significant concern in recent years is forest wildfires. A real-time simulation system for accurately predicting where and how fast a forest fire will spread would assist at the tactical level in effectively controlling it. In modeling and simulating forest fire spread using the cellular discrete event approach, the forest cell model is the basic building block of the entire simulation model. Therefore, it is necessary to express the forest cell model in the discrete event specification (DEVS) formalism being used in order to mathematically verify the model correctness before implementing it. In this paper we present a DEVS forest cell model and express it directly in the Parallel DEVS formalism, thus mathematically verifying the model correctness. We also argue that the forest cell model cannot be directly expressed in the Timed Cell-DEVS formalism due to its requirement for binary states and one time delay per cell.

1 INTRODUCTION

A lot of ecological problems are concerned with propagation processes and therefore, their study often require to develop models that take into account the system evolution in both time and space. Such problems are generally of a large-scale nature and are difficult to efficiently simulate. Furthermore, the models for simulating the processes underlying these ecological problems are spatially distributed and require large amounts of data for simulation. An example of such an ecological problem is forest wildfires.

In recent years forest wildfires have become so destructive throughout the world that it is more likely that this trend will continue. At the *strategic* level, greater attention must be focused on the underlying causes, the effect of land management on fire ecology, wildfire risk, the dynamics of vegetation fuel, and how to reduce the likelihood of catastrophic fires [Andrews and Queen 2000]. However, once a forest wildfire has started, a real-time simulation system for accurately predicting where and how fast the fire will spread would assist at the *tactical* level to effectively bring it under control. Toward that objective, the work reported in [Ntaimo et al., 2003] develops a cellular DEVS model [Zeigler et al., 2000; Zeigler and sarjoughian 2002] of forest fire spread that includes response control measures. The cellular DEVS modeling approach is chosen because of its ability to effectively represent large-scale spatial dynamic phenomena for efficient simulation [Ameghino et al., 2001; Muzy et al., 2002; Zeigler 2003].

In modeling fire spread using the cellular DEVS approach, the forest cell model is the basic building block of the entire simulation model. Therefore, it becomes necessary to express the forest cell model in the DEVS formalism being used in order to mathematically verify the model correctness before implementing it. In this paper we mathematically verify the model correctness of the forest cell model presented in [Ntaimo et al., 2003] by expressing it

directly in Parallel DEVS formalism [Zeigler and sarjoughian 2002]. We also argue that our forest cell model cannot be directly expressed in Timed Cell-DEVS formalism [Wainer and Giambiasi 1998] mainly due to the fact that the forest cell model requires multiple general states and multiple time delays. The Timed Cell-DEVS formalism allows only for binary states and one delay per cell, while the Parallel DEVS formalism allows for multiple general states as well as multiple time delays per cell, both of which are required by our forest cell model.

This paper is organized as follows. In the next section we give a description of the forest cell model and describe how fire spread is modeled in the cell. In addition, we briefly mention how simple fire suppression rules are incorporated in the cell model. In section 3 we express the forest cell model in Parallel DEVS, thus verifying the model correctness mathematically. An attempt to express the forest cell model in Timed Cell-DEVS is made in section 4.2. We end our paper with some concluding remarks and point out some future research directions along this line of work.

2 THE DEVS FOREST CELL MODEL

We now give a description of the forest cell model. We first state the appropriate nomenclature and then describe the cell states, inputs and outputs. We also describe fire spread and fire suppression in the cell. Throughout this paper | is the mathematical symbol for "such that" and || is the symbol for the operator "or". Let us define the following:

$xcoord$: cell's x coordinate in the cellspace;

$ycoord$: cell's y coordinate in the cellspace;

$cell\_id$: cell's id, and $cell\_id = \{xcoord, ycoord \mid xcoord, ycoord \in \mathcal{Z}^+\}$, is a pair

structure where $\mathcal{Z}^+$ is the set of positive integers;

wsp: wind speed;

dir: wind direction;

wsp_dir_pair: cell's wind speed and direction pair, where, $wsp\_dir\_pair = \{wsp, dir \mid wsp,$

$dir \in \Re^+\}$, is the wind speed and direction pair structure;

ros: rate of spread;

sdr: spread direction;

fli: fireline intensity;

T_{fli}: the forest cell's fireline intensity threshold for ignition;

fln: flame length;

$spreadEnt$: cell fire spread entity, where $spreadEnt = \{this.cell\_id, ros, sdr, fli, fln \mid ros,$

$sdr, fli, fln \in \Re^+\}$, is the spread entity.

2.1 Model States, Inputs and Outputs

The forest cell atomic model has 6 basic states; *unburned, unburned_wet, burning, burning_wet, burned*, and *burned_wet*. The cell has an additional state,*weather_update*, which allows for the update of wind speed and direction across the cell.

The the model has eleven basic input ports, namely; "inN", "inNE", "inE", "inSE", "inS", "inSW", "inW", "inNW", "inWeather", "inIgniter", and "inFirefight". The last three input ports allow for the transmission of weather, ignition and fire fighting information, respectively, to the cell. The other input ports allow for the cell to receive input from the neighbor cell as a signal for potential cell ignition if the cell is in the *unburned* state. The cell also has two additional inputs, "start" and "stop", for initialization and stopping the simulation, respectively.

When a cell receives input on any of the input ports "inN" through "inNW", it calls a method to compute its fireline intensity based on the cell's fuel model, slope, aspect, and prevailing wind speed and direction. Let us call this method `computeFlnIntensity()` and we will use it in expressing the forest cell model in the Parallel DEVS formalism. If a cell's fireline intensity is above the threshold T_{fli} the cell immediately transitions into the *burning* state.

The cell model has nine basic output ports, namely; "outN", "outNE", "outE", "outSE", "outS", "outSW", "outW", "outNW", and "outTrans". The cell transmits its cell ID to the neighbor cells via the first eight ports to signal that fire has now reached them. The last output port allows for the transmission of fire spread variable data to a transducer for the experimental frame. Additional output and input ports can be added to the cell model as needed.

2.2 Fire Spread

In this subsection we describe how fire spread is modeled in our DEVS forest cell model. For clarity, we define two methods for computing fire spread

in a cell and use them in the next section in expressing the forest cell model in Parallel DEVS. We abstract and model fire spread as follows. First, a one-dimensional rate of fire spread and direction using a given fire spread mathematical model such as Rothermel's model [Rothermel 1972] is computed. This model takes in as input the forest cell fuel properties, topography, wind speed and direction. Once the major rate of spread and direction are determined, a decomposition algorithm that takes in as input the computed rate of spread and direction and the topography of the cell (mainly slope and aspect), is applied to calculate the two-dimensional rate of spread in the eight major spread directions; N, NE, E, SE, S, SW, W, and NW.

Fire spread is modeled as spreading from the center of an ignited cell towards the neighbor cells. Therefore, in order to simulate the rate of fire spread in the eight directions, the time it takes for a fire spread component to reach the border of the associated neighbor cell is determined. We call this time the *burn delay* for that component. This is calculated by the following simple equation:

$$t_i = \frac{d_i}{R_i}, \tag{1}$$

where t_i is the *burn delay* for spread direction i, d_i is the distance from the center of the cell to the border in the direction i, and R_i is the rate of fire spread in the direction i.

The forest cell model also incorporates dynamism by responding and adapting to changes in weather conditions as the fire spreads. When a cell is in the burning phase it is required that the cell update its rates of spread based on the current wind speed and direction. This is only applicable to the spread components that are still active. Otherwise, no update is necessary. Let us assume that a spread component i has covered a distance d out of a total distance d_i when new wind speed and direction values are input to the cell. At this point the new spread R_i^{new} is computed based on the prevailing cell wind conditions. The remaining delay time t_i^{new} for the spread component to reach the border of the neighbor cell is now computed over the remaining distance $d_i - d$ as follows:

$$t_i^{new} = \frac{d_i - d}{R_i^{new}} \tag{2}$$

Let us now define two methods, `computeFirespread` and `reComputeFirespread` for computing fire spread in a forest cell. We use the first method for computing fire spread and scheduling *burn delays* in a cell that has just been ignited. The second method will be called for updating the *burn delays* for a burning cell when there is a weather update.

The *burn delays* for all the eight components are scheduled for simulation as follows. First, delays are put in nondecreasing order and the minimum delay is subtracted from the rest. The result are the simulation burn delays for each spread direction and are added to an *order container*. At anytime the order container is not empty we can ask for the minimum delay. Thus, the minimum delay is always the scheduled burn delay whenever the order container is not empty. Once the cell transitions into the *burning* state the first scheduled burning delay is removed from the order container and the cell holds in the *burning* state for the duration of the delay. This process is continued while the order container is not empty. Once the last scheduled delay is consumed (*order container* is empty), the cell transitions into the *burned* state.

On the other hand, if while a forest cell is in the *burning* state and there is a weather update (change in wind speed and direction), the burn delays for the active components are no longer valid and need to be updated and rescheduled for simulation. In this case we call the method `reComputeFirespread()` to perform this function. Let us define the order container object `fireSpread`, to store the scheduled *burn delays* with their associated spread component directions. Also, let `fireSpread` have methods `size()`, `getBurndelay()`, and `getBurnDirection()`, for returning the number of delays in the container, the minimum *burn delay* value, and the associated burn delay direction, respectively.

2.3 Fire Suppression

To allow the forest fire cell model to handle fire suppression scenarios, we derive simple fire suppression rules in [Ntaimo et al., 2003] based on flame length or fireline intensity as given by the general reliable rules for fire suppression [Andrews 1986; Rothermel and Rinehard 1983]. We assume that fire suppression in a *burning* cell takes at most T_r units of time, where T_r is the time remaining for that cell to transition into the *burned* state. The actual time for putting out a fire in a cell is usually unknown. Therefore, we define a nonnegative random variable α_s as the fraction of the remaining time for a cell to

transition from *burning* to *burned* at the instant fire suppression is initiated for a fire fighting scenario s. Thus the actual time to put out a fire in the cell is equal to $\alpha_s T_r$.

In our forest cell model we currently consider only the four fire fighting scenarios are that are described in the four general reliable rules for fire suppression [Andrews 1986; Rothermel and Rinehard 1983]. Fire suppression is possible for scenario $s = 1$ when ($fln <$ 1.2 meters) and scenario $s = 2$ when (1.2 $\leq fln <$ 2.4 meters). For scenarios $s = 3$ and $s = 4$, however, fire cannot be put out by any fire fighting efforts, except by prevention only. Therefore, $0 \leq \alpha_s \leq 1$ for $s = 1, 2$, and $\alpha_s > 1$ for $s = 3, 4$. We assume that the random variable α_s is either given or determined from an appropriate probability distribution.

3 EXPRESSING THE FOREST CELL MODEL IN PARALLEL DEVS

In this section we express the forest cell model in Parallel DEVS formalism [Zeigler and Sarjoughian 2002]. We start by stating the formal definition of Parallel DEVS. A basic Parallel DEVS [Zeigler and Sarjoughian 2002] is a structure:

$$DEVS = (X_M, Y_M, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta)$$

where

$X_M = \{(p, v) \mid p \in InPorts, v \in X_p\}$ is the set of input ports and values;

$Y_M = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$ is the set of output ports and values;

S is the set of sequential states;

$\delta_{ext} : Q \times X_M^b \rightarrow S$ is the external state transition function;

$\delta_{int} : S \rightarrow S$ is the internal state transition function;

$\delta_{con} : Q \times X_M^b \rightarrow S$ is the confluent transition function;

$\lambda : S \rightarrow Y^b$ is the output function;

$ta : S \rightarrow R_0^+ \bigcup \infty$ is the time advance function;

with $Q := \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$ the total set of states.

3.1 Model Expression in Parallel DEVS

We are now in a position to express the forest cell atomic model in Parallel DEVS. The forest cell atomic model can be defined in Parallel DEVS as follows:

$$DEVS = (X_M, Y_M, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta)$$

where

InPorts = {"start", "stop", "inN", "inNE", "inE", "inSE", "inS", "inSW", "inW", "inNW", "inWeather", "inIgniter", "inFireFight"}
where

$X_{start} = \{\}(emptyset)$;
$X_{stop} = \{\}(emptyset)$;
$X_{inN} = \{cell\_id\}$;
$X_{inNE} = \{cell\_id\}$;
$X_{inE} = \{cell\_id\}$;
$X_{inSE} = \{cell\_id\}$;
$X_{inS} = \{cell\_id\}$;
$X_{inSW} = \{cell\_id\}$;
$X_{inW} = \{cell\_id\}$;
$X_{inNW} = \{cell\_id\}$;
$X_{inWeather} = \{wsp\_dir\_pair\}$;
$X_{inIgniter} = \{cell\_id\}$;
$X_{inFirefight} = \{cell\_id\}$;

with

$X_M = \{(p, v) \mid p \in InPorts, v \in X_p\}$ is the set of input ports and value pairs.

Outports = {"outN", "outNE", "outE", "outSE", "outS", "outSW", "outW", "outNW", "outTrans"}

where

$Y_{outN} = \{\text{this}.cell\_id\}$;
$Y_{outNE} = \{\text{this}.cell\_id\}$;
$X_{outE} = \{\text{this}.cell\_id\}$;
$Y_{outSE} = \{\text{this}.cell\_id\}$;
$Y_{outS} = \{\text{this}.cell\_id\}$;
$Y_{outSW} = \{\text{this}.cell\_id\}$;
$Y_{outW} = \{\text{this}.cell\_id\}$;
$Y_{outNW} = \{\text{this}.cell\_id\}$;
$Y_{outTrans} = \{spreadEnt\}$;

with

$Y_M = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$ is the set of output ports and value pairs.

In the following *phase* is a control state that is almost always used in models to keep track of where the full state is. Sigma (σ) holds the time remaining to the next internal event. This is precisely the time-advance value to be produced by the time-advance function. The parameter e is the elapsed time in a given state. Also, "in$_i$" and "out$_i$" represent the input and output ports $i \in$ {N, NE, E, SE, S, SW, W, NW} to the neighbor cells.

External Transition Function:

$\delta_{ext}((phase, \sigma, fli, fln), e, (p, v)) =$
("unburned", ∞, fli, fln),
    if $p =$ "start"
($phase$, ∞, fli, fln),
    if $p =$ "stop"
("to_burning", 0, fli, fln),
    if $phase =$ "unburned" & ($p =$ "in$_i$" ||
    $p =$ "inIgniter") & $fli > T_{fli}$
        $fli =$ `computeFlnIntensity()`
("weather_update", 0, fli, fln),
    if $phase =$ "burning" & $p =$ "inWeather"
("to_unburned_wet", 0, fli, fln),
    if $phase =$ "unburned" & $p =$ "inFirefight"
('to_'burning_wet", 0, fli, fln),
    if $phase =$ "burning" & $p =$ "inFirefight" &
    $fln < 2.4$
($phase$, $\sigma - e, fli, fln$)
    otherwise;

Internal Transition Function:

$\delta_{int}((phase, \sigma, fli, fln), e, (p, v)) =$
("burning", t_i, fli, fln),
    if $phase =$ "to_burning"
        `computeFirespread(fireSpread)`
        $i =$ `fireSpread.getBurnDirection()`
        $t_i =$ `fireSpread.getBurndelay()`
("burning", t_i, fli, fln),
    if $phase =$ "burning" & $size > 0$
        $size =$ `fireSpread.size()`
        if ($size > 0$)
            $i =$ `fireSpread.getBurnDirection()`
            $t_i =$ `fireSpread.getBurndelay()`
("burning", t_i, fli, fln),
    if $phase =$ "weather_update" & $size > 0$
        `reComputeFirespread(fireSpread)`
        $size =$ `fireSpread.size()`
        if ($size > 0$)
            $i =$ `fireSpread.getBurnDirection()`
            $t_i =$ `fireSpread.getBurndelay()`
("burned", ∞, fli, fln),
    if $phase =$ "burning" & $size = 0$
        $size =$ `fireSpread.size()`
("burned_wet", ∞, fli, fln),
    if $phase =$ "burning_wet" & ($t_\alpha = 0$)
("burning_wet", t_{α_s}, fli, fln),
    if $phase =$ "to_burning_wet" &
    $p =$ "inFirefight"
        $T_r = \sigma - e$
        if $fli < 1.2$
            s = 1
            Determine α_1
            $t_{\alpha_s} = \alpha_1 T_r$
        else if $fli < 2.4$
            s = 2
            Determine α_2
            $t_{\alpha_s} = \alpha_2 T_r$
("unburned_wet",∞, fli, fln),
    if $phase =$ "to_unburned_wet"

Confluence Function:

$\delta_{con}(s, ta(s), x) = \delta_{int}(\delta_{ext}(s, 0, x))$;

Output Function:

λ(phase, σ, fli, fln) =
(out$_i$, this.$cellid$)
    if $phase =$ ("burning" || "burning_wet") &
    $t_i = 0$
(outTrans, {this.$cell\_id, ros, sdr, fli, fln$})
    if $phase =$ "to_burning"
(outTrans, {this.$cell\_id, ros, sdr, fli, fln$})
    if $phase =$ "weather_update"
$\emptyset$ (null output)
    otherwise;

Time advance Function:

ta(phase, σ, fli, fln) = σ;

The above is the direct expression of the forest cell model in Parallel DEVS and provides a mathematical verification of the model correctness. Next, we

attempt to express the forest cell model directly in Timed Cell-DEVS.

4 EXPRESSING THE FOREST CELL MODEL TIMED CELL-DEVS

In this section we argue that the forest cell atomic model cannot be directly expressed in Timed Cell-DEVS formalism. The Cell-DEVS formalism is formally described in [Wainer and Giambiasi 1998] and implementation models given in [Ameghino et al., 2001], and [Wainer and Giambiasi 2001]. We first state the formal definition of three-states transport delay model for the Timed Cell-DEVS formalism, which is an extension to the basic binary Cell-DEVS model with different delays. To extend to the three-states transport delay model, a third state reflecting undefined behavior for the cell is introduced and is denoted as ϕ, and serves to define the behavior of those cells whose binary state is unknown.

4.1 Timed Cell-DEVS Definition

A three-states transport delay Timed Cell-DEVS model [Wainer and Giambiasi 1998] can be formally defined as:

$$TDCD =< I, X, Y, delay, S, d, \delta int, \delta ext, \tau, \lambda, D >$$

where,

$I =< \eta, P^X, P^Y >$ represents the definition of the modular model interface.

Here $\eta \in \mathbf{N}$ is the neighborhood size, and,

for $i = X|Y, P^i$ is a port definition (input or output respectively), where

$P^i = \{(N_j^i, T_j^i)\ /\ \forall j \in [1, \eta],\ N_j^i \in [i_1, i_\eta]$ (port name), and
T_j^i = binary (port type)$\}$;

$X = \{0, 1, \phi\}$ is the set of input external events;

$Y = \{0, 1, \phi\}$ is the set of output external events;

$delay \in$ {transport, inertial, none};

S is the state set, where

$S = \{$(s, phase, $\sigma queue, \sigma)$ /
$s \in \{0, 1, \phi\}$,
phase $\in$ {active, passive},
$\sigma queue = \{((v_1, \sigma_1), ..., (v_m, \sigma_m))$ / $m \in \mathbf{N} \wedge \forall i \in [1, m], i \in \mathbf{N}$,
$a_i \in \Re_0^+ \bigcup \infty, v_i \in \{0, 1\}\}$, and
$\sigma \in \Re_0^+ \bigcup \infty$
$\}$ if delay = transport, or

S= $\{(s, phase, f, \sigma)$ /
$s \in \{0, 1\}$,
phase $\in$ {active, passive},
$f \in \{0, 1\}$, and
$\sigma \in \Re_0^+ \bigcup \infty$
$\}$ if delay = inertial;

$\mathbf{N} \in \{0, 1, \phi\}^\eta$, is the set of the neighborhood's binary states;

$\mathbf{d} \in \Re_0^+$ is the transport delay for the cell;

$\delta int : S \rightarrow S$ is the internal transition function;

$\delta ext : Q \times X$ is the external transition function, where Q is the state set defined by

$Q = \{(s, e)/s \in S$, and $e \in [0, D(s)]\}$;

$\tau : N \rightarrow N$ is the local computing function;

$\lambda : S \rightarrow Y$ is the output function; and

$D : Q \rightarrow \Re_0^+ \bigcup \infty$ is the state's lifetime function.

4.2 Model Expression in Timed Cell-DEVS

Let us begin our argument with expressing the cell input X in Timed Cell-DEVS. The forest cell inputs X_{start}, X_{stop}, the neighbor cells inputs $X_{inN}, ..., X_{inNW}$, and $X_{inIgniter}$ and $X_{inFirefight}$ can be converted to binary ($X = \{0, 1, \phi\}$) as required by the Timed Cell-DEVS formalism. This is because in all these inputs we simply communicate the cell_id of the neighbor cell. Instead, we can as well send a 1 or 0 to the cell of interest and still achieve the same objective. However, the input $X_{inWeather}$ require that we pass the global wind speed and direction (wsp_dir_pair) to all the cells. Therefore, this input may not be directly expressed in Timed Cell-DEVS. Thus, the forest cell model external transition function cannot be directly expressed in Timed Cell-DEVS.

We can make a similar argument in the limitation of expressing the forest cell model output into $Y = \{0, 1, \phi\}$, which is required by Timed Cell-DEVS. The forest cell output $Y_{outTrans}$ require relatively more complex data types compared to the binary data type. Again, we may not directly express the forest cell model output function λ in the

Timed Cell-DEVS unless we apply some transformation to the data.

The main issue, however, is in expressing the forest cell model internal transition function into that of Timed Cell-DEVS. Here the limitation is in handling multiple states each with possibly multiple delays. As explained in section 2, the forest cell model has 6 basic states, each with a different delay. Furthermore, the *burning* state requires that each burning component delay be scheduled and that the output at the end of the delay be sent to the appropriate neighbor cell, thus allowing for independent fire spread in the 8 major component directions. This state has multiple delays. Besides only allowing for binary states ($s \in \{0, 1, \phi\}$), Timed Cell-DEVS permits only one time delay per cell of type ($\mathbf{d} \in \Re_0^+$). Therefore, the forest cell model cannot be directly expressed in Timed Cell-DEVS. Parallel DEVS has neither of these limitations and thus allows for the direct expression of the forest cell model.

5 CONCLUSION

In this paper a DEVS forest cell model has been expressed in Parallel DEVS formalism and the model correctness thus mathematically verified. In modeling fire spread using the cellular DEVS approach, the forest cell model is the basic building block of the entire simulation model. Therefore, it is necessary to express the forest cell model in the DEVS formalism in order to verify model correctness mathematically before implementing it. Due to the requirement of general states and multiple time delays, we argue that the forest cell model cannot be directly expressed in the Timed Cell-DEVS formalism. Unlike the Parallel DEVS formalism, the Timed Cell-DEVS formalism allows only for models with binary states and one delay per cell.

Our future work along this direction of research include the development and mathematical verification of a Parallel DEVS atomic model for a forest "fire suppression agent" that is going to be coupled to the forest cell model discussed in this paper. In this way, all fire suppression activities for a given forest cell would be modeled via the "agent" for that cell. We believe that this would enable more flexibility in simulating diverse fire suppression scenarios more realistically. Furthermore, this would allow for fire suppression control rules to be separate from the forest cell and thus, would allow for more complex fire suppression activities to be modeled and simulated.

REFERENCES

Ameghino J., A. Troccoli, G. Wainer. 2001. "Models of Complex Physical Systems using Cell-DEVS," In *Proceedings of Annual Simulation Symposium*, Seattle, WA. U.S.A.

Andrews, P.L. 1986. "BEHAVE: Fire Behavior Predictions and Fuel Modeling System-Burn Subsystem Part 1," USDA Forest Service General Technical Report INT-194. 130 p.

Andrews, P.L. and L.P. Queen. 2001. "Fire Modeling and Information System Technology," *International Journal of Wildland Fire*, 10, pp. 343-352.

Muzy, A., G. Wainer, E. Innocenti, A. and Aiello, J.F. Santucci. 2002. "Comparing Simulation Methods for Fire Spreading Across a Fuel Bed," in *Proceedings of AIS'2002*, Lisbon, Portugal.

Ntaimo, L., B. Khargharia, B.P. Zeigler and M.J. Vasconcelos. 2003. "Studying Forest Fire Spread and Suppression in DEVS," submitted for publication. http://www.u.arizona.edu/∼ntaimo/papers/ForestFire.pdf

Rothermel, R. 1972. "A Mathematical Model for Predicting Fire Spread in Wildland Fuels," Research Paper INT-115. Ogden, UT: U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station.

Rothermel, R.C. and G.C. Rinehard. 1983. "Field Procedures for Verification and Adjustment of Fire Behavior Predictions," USDA Forest Service General Technical Report INT-142. Intermountain Forest and Range Experiment Station, Ogden, UT.

Wainer, G. and N. Giambiasi. 1998. "Specification, Modeling and Simulation of Timed Cell-DEVS Spaces," *Technical Report n.: 97-006, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires*, Argentina.

Wainer, G. and N. Giambiasi. 2001. "Timed Cell-DEVS: Modeling and Simulation of Cell Spaces," In *Discrete Event Modeling & Simulation*

Technologies: A Tapestry of Systems and AI-based Theories and Methodologies, Sarjoughian H. S. and F.E. Cellier, Eds., Spinger, pp. 187-213.

Zeigler B. P., H. Praehofer, and T.G. Kim. 2000. *Theory of modeling and simulation*, 2nd Edition, Academic Press.

Zeigler B. P. and H. Sarjoughian. 2002. *Introduction to DEVS Modeling and Simulation with JAVA: A Simplified Approach to HLA-Compliant Distributed Simulations*, The University of Arizona, Tucson, Arizona, USA.

Zeigler, B.P. 2003. *Discrete Event Abstraction: An Emerging Paradigm for Modeling Complex Adaptive Systems, Advances in Adaptive Complex Systems*, edited by L. Booker, Sante Fe Institute/Oxford Press, Oxford (in press).