Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Moench, O. Rose, eds.

VESICLE-SYNAPSIN INTERACTIONS MODELED WITH CELL-DEVS

Rhys Goldstein
Gabriel Wainer

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive, Ottawa, ON, K1S 5B6, Canada

James J. Cheetham
Roderick S. Bain

Department of Biology
Carleton University
1125 Colonel By Drive, Ottawa, ON, K1S 5B6, Canada

ABSTRACT

Interactions between synaptic vesicles and synapsin in a presynaptic nerve terminal were modeled using the Cell-DEVS formalism. Vesicles and synapsins move randomly within the presynaptic compartment. Synapsins can bind to more than one vesicle simultaneously, causing clusters to form. Phosphorylation of synapsin reduces its affinity for vesicles, and causes the clusters to break apart. Upon dephosphosphorylation, new clusters form. Taking advantage of Cell-DEVS, as opposed to traditional techniques for implementing cellular automata, the model prevents collisions between arbitrarily large clusters using transition rules restricted to a 5-cell neighborhood. Simulation results indicate that, in a qualitative sense, the behavior of vesicles and synapsin in neurons was captured.

1 INTRODUCTION

Regulation of neurotransmitter release is important for synaptic plasticity which mediates learning and memory. An important challenge in modern neuroscience is to understand the molecular structures and dynamics underlying regulation of neurotransmitter release. Neurotransmitter release is triggered by the arrival of an action potential in a presynaptic nerve terminal. Within the presynaptic nerve terminal, small synaptic vesicles containing neurotransmitters are present in readily-releasable and reserve pools. Vesicles in the reserve pool are tethered together by synapsins to form clusters. The arriving action potential causes influx of calcium ions through ion channels. Calcium ions trigger biochemical events which result in readily-releasable vesicles releasing neurotransmitters. Calcium also triggers phosphorylation of synapsin by protein kinases. This reduces the affinity of synapsin for vesicles and the cytoskeleton, which disrupts the reserve pool cluster, and allows these vesicles to participate in neurotransmitter release. Thus synapsin can regulate the release of neurotransmitters by controlling the distribution of synaptic vesicles.

Although experimental observations have led to numerous theories describing the relationship between vesicle-synapsin clusters and neurotransmitter release, the process is far from thoroughly understood. Modeling and simulation is frequently applied to the study of complex biological processes, and may well aid the investigation of vesicles and synapsins in a presynaptic nerve terminal. Approaching this task from a software engineering perspective, we have developed a cellular model that captures the motion of vesicle-synapsin clusters and their interaction with the nerve cell membrane. Our work extends a preexisting model (Wainer et al. 2007), which captured the formation and disruption of clusters.

Departing from traditional cellular automata implemented with fixed time increments, our work takes advantage of the Cell-DEVS formalism (Wainer and Giambiasi 2002). This methodology was found to be particularly useful for describing the random motion of vesicle-synapsin clusters. A traditional approach would typically require an algorithm to process the entire cell-space in order to avoid collisions between clusters. Using the timed events accommodated by Cell-DEVS, we demonstrate how collisions between arbitrarily large structures can be avoided with transition rules restricted to a 5-cell neighborhood.

First, an overview of Cell-DEVS is provided, followed by an informal description of the model. The model specification, discussed in Sections 4 to 7, demonstrates the use of Cell-DEVS and presents a formal description of the cluster motion algorithm mentioned above. The model was implemented using the CD++ toolkit, and as discussed in Section 8, test results indicate that the desired qualitative behavior of vesicles and synapsins was captured.

2 CELL-DEVS AND CD++

The DEVS formalism (Zeigler, Kim, and Praehofer 2000) provides a framework for the construction of hierarchical modular models, allowing for model reuse, and reducing development and testing times. Basic models, called “atomic models", are specified through transition functions. Multiple DEVS models can be integrated together to form hierarchical structural models, called "coupled models".

The Cell-DEVS formalism was defined as an extension to cellular automata (Wolfram 2002) combined with DEVS. In Cell-DEVS, each cell in a cellular model is seen as a DEVS atomic model, and a procedure for coupling cells is defined based on the neighborhood relationship. Only the active cells in the cell space are triggered, independently from any activation period. Each cell of a Cell-DEVS model holds state variables and a local computing function, which updates the cell state by using its present state and its neighborhood.

A timed DEVS cell atomic model, associated with each cell in a cellular model, is specified as follows:

$$TDC = \langle X, Y, S, N, delay, d, \delta_{ext}, \delta_{int}, \tau, \lambda, D \rangle$$

The variable X defines the external inputs, Y defines the external outputs, and S is the cell state definition. The variable N represents the set of relative coordinates of each cell in the neighborhood. The $delay$ is the kind of delay used for the cell, and d is its duration. A $transport$ delay can be associated with each cell, which defers the outputs for the cell. A state change will be discarded if it is not steady during an $inertial$ delay. The local computing function τ is used to evaluate the future state of the cell. The remaining functions drive the cell's behavior: δ_{int} for internal transitions, δ_{ext} for external transitions, λ for outputs, and D for the state's duration.

A Cell-DEVS coupled model, representing an entire cell space, is specified as follows:

$$GCC = \langle X_{list}, Y_{list}, X, Y, n, [t_1, \ldots, t_n], N, C, B, Z \rangle$$

Here, X_{list} and Y_{list} are input/output coupling lists, used to define the model's interface. X and Y represent the input/output events. The n value defines the number of dimensions of the cell space, and $[t_1, \ldots, t_n]$ is the number of cells in each dimension. N is the neighborhood set. The cell space is defined by C, together with B, the set of border cells, and Z, the translation function.

CD++ (Wainer 2002) is a modeling tool that implements the DEVS and Cell-DEVS formalisms. DEVS atomic models are programmed in C++, while both DEVS coupled models and Cell-DEVS models can be defined using a built-in specification language. CD++ makes use of the independence between modeling and simulation provided by DEVS, and different simulation engines have been defined for the platform.

3 MODEL DESCRIPTION

In-depth discussions on the chemical interactions that take place at a presynaptic nerve terminal can be found elsewhere (Turner, Burgoyne, and Morgan (1999) and Fdez and Hilfiker (2006), for example). At the risk of adopting an over-simplistic interpretation of an extraordinarily complex system, we will describe a presynaptic terminal as a spherical compartment containing mobile vesicles and synapsins. The compartment is surrounded by a membrane, and we will refer to one connected region in that membrane as the "active zone". It is at the active zone that readily-releasable vesicles fuse and release contained neurotransmitters to the extracellular fluid in response to a signal, or "action potential".

Synapsins move freely through the compartment, binding with vesicles to form clusters. In the model, an individual synapsin may bind with a most two vesicles, though a vesicle may bind with more than two synapsins. When an action potential arrives, the resulting influx of calcium ions ultimately causes some of the bound vesicles and synapsins to separate from one another. After the action potential passes, the disrupted clusters begin to reform.

In the original cellular model (Wainer et al. 2007), the locations of vesicles and synapsins were represented by single cells in a 2-dimensional cell-space. Isolated vesicles and synapsins would move randomly through the space. After each cycle, they would bind to one another to form stationary clusters. Arbitrary probabilities controlled the binding of vesicles and synapsins, as well as their separation from clusters.

Our enhancements include the introduction of action potentials, the nerve cell membrane, the active zone, and the motion of clusters. A reaction triggered by an action potential is modeled as a period of time during which the binding and separation of vesicles and synapsins occur with an alternative set of probabilities. The membrane introduces a circular boundary that vesicles and synapsins cannot cross. The active zone is modeled as a region of the membrane adjacent to which any vesicle or synapsin is rendered motionless. Instead of requiring that clusters remain motionless in general, as in the preexisting model, they are now allowed to move randomly throughout the compartment.

4 MODEL SPECIFICATION

The tuple below is a Cell-DEVS coupled model that represents a presynaptic nerve terminal. The parameter R, which must be a positive integer, is the inner radius of the terminal. The size of the active zone is described by the angle θ, and the probabilities p_V and p_S describe the initial distribution of vesicles and synapsins. In the absence of an action potential, the probability that a vesicle will bind to an adjacent synapsin is p_{rest}. If they are already bound, then there is a probability of q_{rest} that they will separate. The binding and separation probabilities during an action potential are p_{act} and q_{act} respectively. As action potentials induce the separation of clusters, one expects $p_{rest} > p_{act}$ and $q_{rest} < q_{act}$.

$$presynaptic(R, \theta, p_V, p_S, p_{rest}, q_{rest}, p_{act}, q_{act}) = \\ \langle X_{list}, Y_{list}, X, Y, n, [t_1, t_2], N, C, B, Z \rangle$$

The cell-space is a 2-dimensional square grid ($n = 2$), just large enough to surround the inner terminal with a membrane layer of at least one cell in any direction ($t_1 = t_2 = 2 \cdot R + 1$).

There are no outputs ($Y_{list} = Y = \varnothing$), but the model has one input port, named *in*, with which to receive a value of either $receiving_{act}$ or $receiving_{rest}$. These values represent, respectively, the arrival of an action potential from the axon of the neuron, or the subsiding of the reaction triggered by the action potential. Arbitrarily, the cell $[0,0]$ was chosen to receive these inputs ($X_{list} = \{[0,0]\}$).

$$X = \{[in, \Phi'] \mid \Phi' \in \{receiving_{act}, receiving_{rest}\}\}$$

A 5-cell von Neumann neighborhood is used.

$$N = \{[0,0],[1,0],[0,1],[-1,0],[0,-1]\}$$

Because vesicles and synapsins never reach the actual model boundary, it is safe and convenient to use a wrapped border ($B = \varnothing$). The translation function Z is defined by the Cell-DEVS formalism, while the timed DEVS cell atomic model is defined by the tuple below for all absolute coordinates $[i_1, i_2]$ in the cell-space.

$$C([i_1,i_2]) = \langle X, Y, S, N, delay, d, \delta_{ext}, \delta_{int}, \tau, \lambda, D \rangle$$

A cell's state must belong to the set S. It is described by seven variables: $type_0$, b_0, Φ_0, ϕ_0, v_0, z_0, and σ_0.

$$S = \{[type_0, b_0, \Phi_0, \phi_0, v_0, z_0, \sigma_0]\}$$

The $type_0$ variable indicates whether a cell is part of the *empty* region inside the terminal, a *vesicle*, a *synapsin*, part of the neuron's *membrane*, or part of the active *zone* within the membrane.

$$type_0 \in \{empty, vesicle, synapsin, membrane, zone\}$$

The function b_0 is defined for the four vectors pointing from a cell to its adjacent neighbors. For each of these directions, the function result can be either *free*, *seeking*, *unseeking*, *looking*, or *binding*. This information facilitates the modeling of vesicle-synapsin binding and separation.

$$b_0([i,j]) \in \{free, seeking, unseeking, looking, binding\}$$

$$[i,j] \in \{[1,0],[0,1],[-1,0],[0,-1]\}$$

The variable Φ_0 indicates whether the cell has received an event external to the coupled model ($receiving_{[...]}$), is starting or ending a reaction induced by an action potential ($starting_{[...]}$), or is waiting for a change ($holding_{[...]}$).

$$\Phi_0 \in \{ \\ receiving_{act}, receiving_{rest}, \\ starting_{act}, starting_{rest}, \\ holding_{act}, holding_{rest}\}$$

There are eight phases, which are cycled through in succession each time vesicles and synapsins move. The current phase is indicated by the variable ϕ_0.

$$\phi_0 \in \{ \\ starting, holding, \\ selecting, binding_S, binding_V, \\ aiming, steering, moving\}$$

The vector v_0 represents the direction in which a vesicle or synapsin is intending to move ($v_0 \in N$), while the priority number z_0 ($0 \leq z_0 \leq 1$) is used to resolve conflicts between moving vesicles and synapsins.

The variable σ_0 represents the time remaining until the next internal transition ($\sigma_0 \geq 0$). This value is therefore the result of D, the time advance function ($D(s) = \sigma_0$). As the model uses inertial delays ($delay = inertial$), the transition indicated by σ_0 may be interrupted by a change in a neighboring cell. The duration d is not used.

The external transition function δ_{ext}, internal transition function δ_{int}, and output function λ are defined as in the Cell-DEVS formalism. The local computing function τ is described in Section 6.

5 INITIAL CONDITIONS

The *type* is the only state variable that depends initially on a cell's absolute coordinates. The radius R can be used to partition the cell-space into a region inside the terminal, and a bordering region representing the membrane of the neuron. On the outside, all cells encompassed by the angle θ are part of the active zone. Otherwise they are regular membrane cells. On the inside, each cell has a probability p_V of being a vesicle, and a probability p_S of being a synapsin. Otherwise the cell is empty.

The initial type of a cell at $[i_1, i_2]$ is given by the $type_{init}$ function. In its definition below, the function *uniform* is used to obtain random values uniformly distributed between

0 and 1. Figure 1 shows one possible initial configuration of cell types.

$$type_{init}([i_1,i_2]) = \begin{pmatrix} r \geq R & \rightarrow type_{outer} \\ r < R & \rightarrow type_{inner} \end{pmatrix}$$

$$type_{outer} = \begin{pmatrix} i_1 - R > r \cdot cos(\frac{\theta}{2}) & \rightarrow zone \\ i_1 - R \leq r \cdot cos(\frac{\theta}{2}) & \rightarrow membrane \end{pmatrix}$$

$$type_{inner} = \begin{pmatrix} U < p_V & \rightarrow vesicle \\ p_V \leq U < p_V + p_S & \rightarrow synapsin \\ p_V + p_S \leq U & \rightarrow empty \end{pmatrix}$$

$$r = \sqrt{(i_1 - R)^2 + (i_2 - R)^2}$$

$$U = uniform()$$

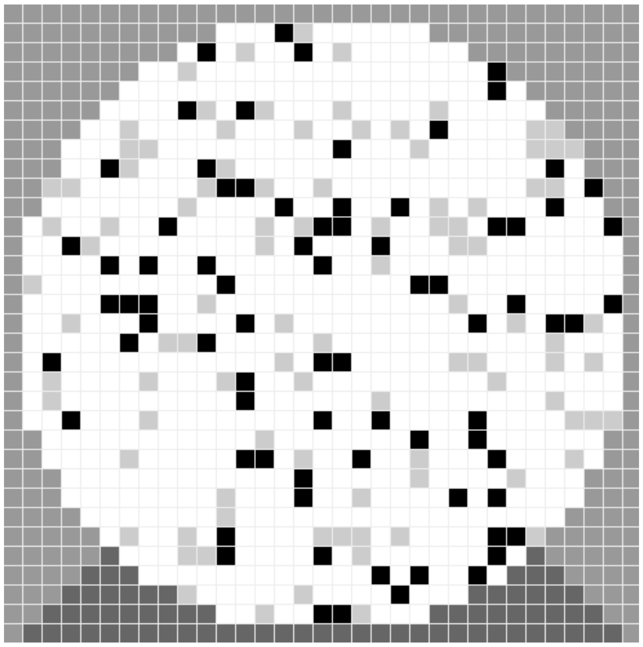

Figure 1: An initial cell-space configuration is shown based on the parameters $R = 16$, $\theta = 90°$, $p_V = 9\%$, and $p_S = 12\%$. On the inside, black cells represent vesicles, light gray cells are synapsins, and white cells are empty. On the outside, the dark region at the bottom is the active zone, while the remainder is the normal membrane. Though it is not part of the model, one could imagine a connection to the axon of the neuron somewhere near the top.

The function b_0 initially results in $free$ regardless of position and direction, representing the absence of vesicle-synapsin clusters ($b_0([i,j]) = free$). The terminal is not initially undergoing a reaction induced by an action potential ($\Phi_0 = holding_{rest}$). The initial phase is the first in the cycle ($\phi_0 = starting$), and the remaining variables are zeroed ($v_0 = [0,0], z_0 = \sigma_0 = 0$).

6 TRANSITION RULES

In order to specify the transition to be made by a cell, a few concepts must be formalized. The term "state" refers to the value associated with each cell in the cell space. A state must be a value in the set S. A "state function" is a function that takes a pair of relative coordinates as an input, and results in the state of the cell associated with those coordinates. The state function s therefore maps values in the set N to values in the set S (for all $[i,j] \in N$, $s([i,j]) \in S$). The expression $s([1,0])$, for example, represents the state of the cell with coordinates $[1,0]$ relative to the cell making the transition.

The functions $type$, b, Φ, ϕ, v, z, and σ are defined implicitly by the equation below. It will be assumed that they are all available in whatever context the state function s is available. The expression $z([1,0])$, for example, gives the priority number of the cell with coordinates $[1,0]$ relative to the transitioning cell.

$$s([i,j]) = [type([i,j]), b([i,j]), \Phi([i,j]), \phi([i,j]), \\ v([i,j]), z([i,j]), \sigma([i,j])]$$

For convenience, the variables $type_0$, b_0, Φ_0, ϕ_0, v_0, z_0, and σ_0 will be used to represent the state of the transitioning cell. The expression Φ_0, for example, will be used as an alternative to $\Phi([0,0])$.

$$s([0,0]) = [type_0, b_0, \Phi_0, \phi_0, v_0, z_0, \sigma_0]$$

The local computing function τ maps the state function s, which carries the present states of neighboring cells, onto the new state of the transitioning cell. It is divided into eight simpler phase-specific functions ($\tau_{starting}$, $\tau_{holding}$, etc.).

$$\tau(s) = \begin{pmatrix} \phi_0 = starting & \rightarrow \tau_{starting}(s) \\ \phi_0 = holding & \rightarrow \tau_{holding}(s) \\ \phi_0 = selecting & \rightarrow \tau_{selecting}(s) \\ \phi_0 = binding_S & \rightarrow \tau_{binding_S}(s) \\ \phi_0 = binding_V & \rightarrow \tau_{binding_V}(s) \\ \phi_0 = aiming & \rightarrow \tau_{aiming}(s) \\ \phi_0 = steering & \rightarrow \tau_{steering}(s) \\ \phi_0 = moving & \rightarrow \tau_{moving}(s) \end{pmatrix}$$

The functions $\tau_{starting}$ and $\tau_{holding}$ ensure that external inputs, representing action potentials, are propagated to every cell. At the completion of the $holding$ phase, the Φ_0 state variable is identical throughout the cell-space. The value is either $holding_{act}$, indicating that a reaction triggered by an action potential is underway, or $holding_{rest}$, indicating the absence of such a reaction.

The functions $\tau_{selecting}$, $\tau_{binding_S}$, and $\tau_{binding_V}$ are responsible for the binding and separation of vesicles and synapsins. It is in these functions that the probabilities p_{rest},

q_{rest}, p_{act}, and q_{act} come into play. If $\Phi_0 = holding_{rest}$, then any adjacent but unbound vesicle-synapsin pair has a probability p_{rest} of becoming bound, and any bound pair has a probability q_{rest} of separating. If $\Phi_0 = holding_{act}$, then p_{act} and q_{act} are used instead.

At the completion of the $binding_V$ phase, every value of every binding function b_0 is either $free$ or $binding$. Also, if any cell is $binding$ in some direction, then the adjacent cell to which it is bound must be $binding$ in the opposite direction.

The rules of binding are complicated by the fact that synapsins can only bind along one axis or the other. The formal specification of these rules is omitted, though a simplified version can be found in the preexisting vesicle-synapsin model (Wainer et al. 2007). Of greater interest in the present model are the rules of cluster motion, captured by the functions τ_{aiming}, $\tau_{steering}$, and τ_{moving}. The formal specification of these functions is given in Section 7

A few additional functions are defined here for convenience. The function any_{type} indicates whether any of the adjacent cells are of the indicated type. Similarly, the function any_{ϕ} indicates whether any of the adjacent cells have the indicated phase.

$$\begin{aligned}&any_{type}(s, type') = \\ &\quad (type([1,0]) = type') \vee (type([-1,0]) = type') \vee \\ &\quad (type([0,1]) = type') \vee (type([0,-1]) = type')\end{aligned}$$

$$\begin{aligned}&any_{\phi}(s, \phi') = \\ &\quad (\phi([1,0]) = \phi') \vee (\phi([-1,0]) = \phi') \vee \\ &\quad (\phi([0,1]) = \phi') \vee (\phi([0,-1]) = \phi')\end{aligned}$$

The function $random$ takes a vector argument, and results in a value randomly selected from that vector.

7 CLUSTER MOTION

While it would be relatively straightforward to allow isolated single-cell particles to move randomly through the presynaptic nerve terminal, it is challenging to specify the motion of vesicle-synapsin clusters. A cluster is any group of vesicles and synapsins connected through $binding$ links defined by the b_0 functions. Clusters move randomly, remaining intact and avoiding obstacles such as other clusters. The algorithm designed to accomplish this is based on priority numbers.

Upon transitioning to the $aiming$ phase ($\phi_0 = aiming$), the direction v_0 and priority number z_0 have been initialized. If a cell is a vesicle or synapsin, and if it is not adjacent to the active zone, then both the direction and priority number are randomized. Otherwise the direction is $[0,0]$, indicating a motionless cell. In this case, the priority number given to empty cells is 1, which is the weakest number. If the motionless cell is not empty, the priority number is zero, which is the strongest.

$$[v_0, z_0] = \begin{pmatrix} is_{movable} & \rightarrow [v_{random}, z_{random}] \\ \neg is_{movable} & \rightarrow [[0,0], z_{frozen}] \end{pmatrix}$$

$$\begin{aligned}&is_{movable} = \\ &\quad (type_0 \in \{vesicle, synapsin\}) \wedge \\ &\quad \neg any_{type}(s, zone)\end{aligned}$$

$$v_{random}() = random([[1,0],[0,1],[-1,0],[0,-1]])$$

$$z_{random}() = uniform()$$

$$z_{frozen}(s) = \begin{pmatrix} type_0 = empty & \rightarrow 1 \\ type_0 \neq empty & \rightarrow 0 \end{pmatrix}$$

The first condition to be specified in the τ_{aiming} function is the last to take effect. If any cell has any neighbors that have already advanced from the $aiming$ phase to the $steering$ phase, that cell must itself advance immediately. Without this condition, the intended transition to the $steering$ phase after 1 time unit could be interrupted, and consequently postponed.

$$\tau_{aiming}(s) = \begin{pmatrix} any_{\phi}(s, \phi'_0) & \rightarrow s'_{now} \\ \neg any_{\phi}(s, \phi'_0) & \rightarrow \tau'_{aiming}(s) \end{pmatrix}$$

$$s'_{now} = [type_0, b_0, \Phi_0, \phi'_0, v_0, z_0, \sigma'_0]$$

$$\phi'_0 = steering$$

$$\sigma'_0 = 0$$

During the $aiming$ phase, directions and priority numbers are repeatedly shared within each cluster. A vesicle or synapsin will adopt these values from an adjacent neighbor to which it is bound, provided that the neighbor has a priority number lower than its own. The process ends when each vesicle and synapsin has the same direction and priority number as any other cell in the same cluster.

It is useful to define a function g_{aiming}, which results in a truthful value if a neighbor at $[i,j]$ has a lower priority number and is bound to the cell. Another function, G_{aiming}, is truthful if all neighbors have advanced past the preceding $binding_V$ phase, and $g_{aiming}([i,j])$ is true for any adjacent cell $[i,j]$.

$$g_{aiming}(s, [i,j]) = (z([i,j]) < z_0) \wedge (b_0([i,j]) = binding)$$

$$\begin{aligned}&G_{aiming}(s) = \neg any_{\phi}(s, binding_V) \wedge (\\ &\quad g_{aiming}(s,[1,0]) \vee g_{aiming}(s,[-1,0]) \vee \\ &\quad g_{aiming}(s,[0,1]) \vee g_{aiming}(s,[0,-1]))\end{aligned}$$

So long as the result of G_{aiming} is false, there is nothing to do other than transition to the *steering* phase after 1 time unit.

$$\tau'_{aiming}(s) = \begin{pmatrix} G_{aiming}(s) & \rightarrow \tau''_{aiming}(s) \\ \neg G_{aiming}(s) & \rightarrow s'_{later} \end{pmatrix}$$

$$s'_{later} = [type_0, b_0, \Phi_0, \phi'_0, v_0, z_0, \sigma'_0]$$

$$\phi'_0 = steering$$

$$\sigma'_0 = 1$$

If $G_{aiming}(s)$ is true, the lower priority number and direction are copied from the adjacent neighbor.

$$\tau''_{aiming}(s) = \begin{pmatrix} g_{aiming}(s,[1,0]) & \rightarrow obey(s,[1,0]) \\ g_{aiming}(s,[0,1]) & \rightarrow obey(s,[0,1]) \\ g_{aiming}(s,[-1,0]) & \rightarrow obey(s,[-1,0]) \\ g_{aiming}(s,[0,-1]) & \rightarrow obey(s,[0,-1]) \end{pmatrix}$$

The copying is specified by the *obey* function below. The time advance of zero ensures that directions and priorities are repeatedly copied until G_{aiming}(s) becomes and remains false.

$$obey(s,[i,j]) = [type_0, b_0, \Phi_0, \phi_0, v([i,j]), z([i,j]), \sigma'_0]$$

$$\sigma'_0 = 0$$

By the time the first cell transitions beyond the *aiming* phase, each cluster has a single direction. That direction may change, however, as clusters must neither collide nor enter the membrane. All possible collisions are to be resolved during the *steering* phase.

The definition of $\tau_{steering}$ is similar to that of τ_{aiming}, but more complex. First, cells are to transition to the *moving* phase if any adjacent neighbors have already done so.

$$\tau_{steering}(s) = \begin{pmatrix} any_\phi(s,\phi'_0) & \rightarrow s'_{now} \\ \neg any_\phi(s,\phi'_0) & \rightarrow \tau'_{steering}(s) \end{pmatrix}$$

$$s'_{now} = [type_0, b_0, \Phi_0, \phi'_0, v_0, z_0, \sigma'_0]$$

$$\phi'_0 = moving$$

$$\sigma'_0 = 0$$

As was the case in the *aiming* phase, a direction and priority number are copied from an adjacent cell only if the priority number is lower. In the case of *steering*, any one of three conditions must also be met. One of those conditions is the same as in the case of *aiming*; specifically, that the transitioning cell is bound to the adjacent neighbor.

The second condition is that the cell is either a vesicle or a synapsin, and is currently intending to move towards the adjacent neighbor with the lower priority number. This prevents a particle from entering a cell that might not otherwise be empty.

The third condition is that the adjacent neighbor is a vesicle or synapsin heading towards the transitioning cell. This condition is relevant even if the transitioning cell is empty. Empty cells take on priorities and directions to prevent two clusters from approaching simultaneously from different directions.

The conditions described above are defined formally in the function $g_{steering}$.

$$\begin{aligned} & g_{steering}(s,[i,j]) = \\ & \quad (z([i,j]) < z_0) \wedge (\\ & \qquad (b_0([i,j]) = binding) \vee \\ & \qquad ((type_0 \in \{vesicle, synapsin\}) \wedge \\ & \qquad\quad (v_0 = [i,j])) \vee \\ & \qquad ((type([i,j]) \in \{vesicle, synapsin\}) \wedge \\ & \qquad\quad (v([i,j]) = [-i,-j]))) \end{aligned}$$

Suppose that an empty cell has an approaching vesicle on either side. Suppose also that the one on the right has a lower priority number. The empty cell in the middle will adopt the direction and priority number from the right. In this case, the direction is pointing to the left. The vesicle on the left will then adopt this lower priority number as well, and reverse its direction. This example illustrates how a possible collision is avoided.

Continuing the example above, suppose that the vesicle on the right in now pushed upwards by a synapsin with a lower priority number. The empty cell now has no approaching particles. In order to reset its direction and priority number, this condition is checked using the function $\gamma_{steering}$.

$$\begin{aligned} & \gamma_{steering}(s,[i,j]) = \\ & \quad (type_0 = empty) \wedge \\ & \quad (v_0 = [-i,-j]) \wedge \\ & \quad (v([i,j]) \neq [-i,-j]) \end{aligned}$$

All the conditions described above are combined into the single function $G_{steering}$.

$$\begin{aligned} & G_{steering}(s) = \neg any_\phi(s, aiming) \wedge (\\ & \quad g_{steering}(s,[1,0]) \vee g_{steering}(s,[-1,0]) \vee \\ & \quad g_{steering}(s,[0,1]) \vee g_{steering}(s,[0,-1]) \vee \\ & \quad \gamma_{steering}(s,[1,0]) \vee \gamma_{steering}(s,[-1,0]) \vee \\ & \quad \gamma_{steering}(s,[0,1]) \vee \gamma_{steering}(s,[0,-1])) \end{aligned}$$

As long as $G_{steering}(s)$ is false, a cell has nothing to do except wait 1 time unit then transition to the *moving* phase.

$$\tau'_{steering}(s) = \begin{pmatrix} G_{steering}(s) & \rightarrow \tau''_{steering}(s) \\ \neg G_{steering}(s) & \rightarrow s'_{later} \end{pmatrix}$$

$$s'_{later} = [type_0, b_0, \Phi_0, \phi'_0, v_0, z_0, \sigma'_0]$$

$$\phi'_0 = moving$$

$$\sigma'_0 = 1$$

If $G_{steering}(s)$ is true, then one of the conditions that requires a change of direction and priority is also true. These possible changes are specified below. The function *obey* is the same as in the *aiming* phase, while $obey_{all}$ resets empty cells.

$$\tau''_{steering}(s) = \begin{pmatrix} g_{steering}(s,[1,0]) & \rightarrow obey(s,[1,0]) \\ g_{steering}(s,[0,1]) & \rightarrow obey(s,[0,1]) \\ g_{steering}(s,[-1,0]) & \rightarrow obey(s,[-1,0]) \\ g_{steering}(s,[0,-1]) & \rightarrow obey(s,[0,-1]) \\ \gamma_{steering}(s,[1,0]) & \rightarrow obey_{all} \\ \gamma_{steering}(s,[0,1]) & \rightarrow obey_{all} \\ \gamma_{steering}(s,[-1,0]) & \rightarrow obey_{all} \\ \gamma_{steering}(s,[0,-1]) & \rightarrow obey_{all} \end{pmatrix}$$

$$obey_{all} = [type_0, b_0, \Phi_0, \phi_0, v'_0, z'_0, \sigma'_0]$$

$$v'_0 = [0,0]$$

$$z'_0 = 1$$

$$\sigma'_0 = 0$$

One might note that the *aiming* phase is somewhat redundant, as the propagation of directions and priorities within a cluster is also performed in the *steering* phase. The inclusion of the *aiming* phase is certain to improve computational efficiency. Also, it is likely to reduce any directional bias in cluster motion that results from the non-deterministic nature of the *steering* algorithm. While the *aiming* phase merely selects the lowest priority number in each cluster, the final state of the cell-space after *steering* depends on the order in which simultaneous transitions are evaluated.

The final phase in the cycle is *moving*, during which vesicles and synapsins translate according to their directions. As ensured in the *steering* phase, these directions will not break bindings and will not cause collisions. Any empty cell about to receive a vesicle or synapsin from one direction has a v_0 pointing in the other direction.

Four of the state variables are assigned as follows.

$$[\phi_m, v_m, z_m, \sigma_m] = [starting, [0,0], 0, 1]$$

The function g_{moving} indicates whether a cell has an incoming particle in a given direction. The function G_{moving} indicates whether a cell has an incoming particle in any direction.

$$\begin{aligned} & g_{moving}(s,[i,j]) = \\ & \quad (type([i,j]) \in \{vesicle, synapsin\}) \wedge \\ & \quad (v([i,j]) = [-i,-j]) \end{aligned}$$

$$\begin{aligned} & G_{moving}(s) = \\ & \quad g_{moving}(s,[1,0]) \vee g_{moving}(s,[-1,0]) \vee \\ & \quad g_{moving}(s,[0,1]) \vee g_{moving}(s,[0,-1]) \end{aligned}$$

If a cell has an incoming particle, τ'_{moving} is evaluated. Otherwise, its future state depends on a function named *move*.

$$\tau_{moving}(s) = \begin{pmatrix} G_{moving}(s) & \rightarrow \tau'_{moving}(s) \\ \neg G_{moving}(s) & \rightarrow move(s) \end{pmatrix}$$

The τ'_{moving} function obtains the $type_0$ and b_0 values from the cell with the incoming vesicle or synapsin. The transition occurs after 1 time unit.

$$\tau'_{moving}(s) = \begin{pmatrix} g_{moving}(s,[1,0]) & \rightarrow from([1,0]) \\ g_{moving}(s,[0,1]) & \rightarrow from([0,1]) \\ g_{moving}(s,[-1,0]) & \rightarrow from([-1,0]) \\ g_{moving}(s,[0,-1]) & \rightarrow from([0,-1]) \end{pmatrix}$$

$$from([i,j]) = [type([i,j]), b([i,j]), \Phi_0, \phi_m, v_m, z_m, \sigma_m]$$

If there are no incoming particles, the one condition to check is whether the cell represents a vacating vesicle or synapsin. If so, the cell becomes empty. Otherwise, its type remains as is.

$$move(s) = \begin{pmatrix} vacating & \rightarrow vacated \\ \neg vacating & \rightarrow remains \end{pmatrix}$$

$$\begin{aligned} & vacating = \\ & \quad (type_0 \in \{vesicle, synapsin\}) \wedge (v_0 \neq [0,0]) \end{aligned}$$

$$vacated = [type'_0, b'_0, \Phi_0, \phi_m, v_m, z_m, \sigma_m]$$

$$remains = [type_0, b_0, \Phi_0, \phi_m, v_m, z_m, \sigma_m]$$

$$type'_0 = empty$$

$$b'_0([i,j]) = free$$

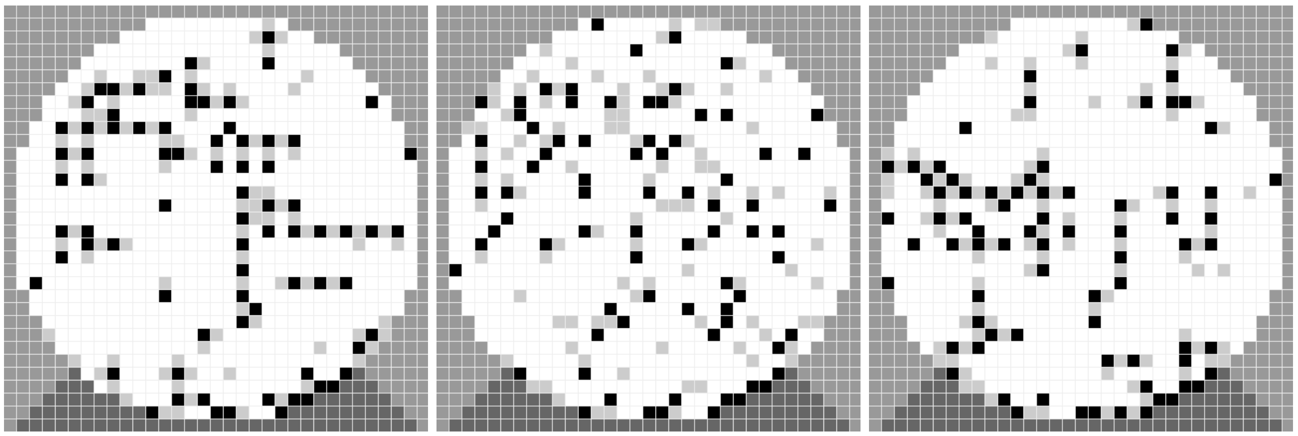

Figure 2: Three snapshots are shown from the test described in the text. The one on the left shows clusters formed after 75 cycles. The first action potential arrived immediately after, and the resulting reaction lasted 5 cycles. Immediately after these 5 cycles, as shown in the center image, the vesicles and synapsins in the large clusters dispersed. Different clusters reformed, as shown on the right, after an additional 75 cycles. Black cells represent vesicles, while light gray cells are synapsins. The active zone, where vesicles and synapsins become bound to the membrane, is at the bottom. The parameters used were as follows: $R = 16$, $\theta = 90°$, $p_V = 9\%$, $p_S = 12\%$, $p_{rest} = 100\%$, $q_{rest} = 1\%$, $p_{act} = 10\%$, and $q_{act} = 50\%$.

8 IMPLEMENTATION AND TESTING

The specified model was implemented using the CD++ toolkit, a task that involved expressing the mathematical formulas in the built-in specification language. The rules below, for example, implement part of the τ'_{moving} function.

```
rule : { #Macro(from_south) }
         1
       { #Macro(is_moving) and
         #Macro(south_is_particle) and
         #Macro(south_is_moving_north) }

rule : { #Macro(from_east) }
         1
       { #Macro(is_moving) and
         #Macro(east_is_particle) and
         #Macro(east_is_moving_west) }
```

The macros `from_south` and `from_east` correspond to the expressions $from([1,0])$ and $from([0,1])$. The number 1, on the second line of each rule, is the time value σ_m. Following this are the conditions that must be satisfied for either rule to take effect. In the first rule, the conditions require $\phi_0 = moving$ and $g_{moving}(s,[1,0])$.

While the specification defines a single Cell-DEVS coupled model, the implementation also included an "axon" model to provide external events at regular intervals. The axon model was given two parameters: one to represent the number of cycles between action potentials, and one to specify the duration of each reaction triggered by an action potential. A "cycle" is a complete rotation through each of the eight phases, a time period during which vesicles and synapsins can move at most once. For convenience, the axon model was itself defined as a Cell-DEVS coupled model containing a single cell. A DEVS atomic model could have been implemented instead.

Tests demonstrated that the model captures the desired qualitative behavior of vesicles and synapsins. The results of one such test are shown in Figure 2. At the beginning of the simulation, vesicles and synapsins were randomly distributed in the terminal. Although this initial condition is not realistic, clusters began forming within the first few cycles. As intended, the clusters mostly broke up after the arrival of an action potential, but regrouped thereafter.

Clusters are smaller and more numerous than they appear in the snapshots of Figure 2. Noting that synapsins may bind to at most two vesicles, and that the vesicles must be opposite one another, one can identify groups of adjacent clusters that at first appear as single larger clusters.

The tendency for clusters to form in straight lines was not intended, and it is unclear why that trend emerges. A greater value of the separation probability q_{rest} would likely result in rounder clusters. Were that the case, linear clusters would be rendered unstable by the fact that a single vesicle-synapsin separation would suffice to sever them.

9 CONCLUSION

A model of vesicle-synapsin interactions was specified using the Cell-DEVS formalism and implemented with CD++. Simulations demonstrated that the model captured the desired qualitative behavior of vesicles and synapsins in presynaptic nerve terminals. Specifically, test results showed the formation and break-up of clusters in response to action potentials, the random motion of clusters, and the docking of vesicles and synapsins near the active zone.

The Cell-DEVS formalism proved particularly useful in describing the motion of vesicle-synapsin clusters. As the algorithm relied on instantaneous transitions, it would have been more cumbersome to implement using a traditional approach to cellular automata with fixed time increments. With Cell-DEVS, it was possible to avoid collisions between arbitrarily large clusters using a neighborhood of only five cells.

Possible future enhancements include the adaptation of the model to a 3-dimensional cell-space, and the fusion and formation of vesicles on the membrane. Binding probabilities and other parameters may be further optimized based on theory or experimental observation. Another challenge would be the representation of long structures known as "actin", which are believed to influence vesicle-synapsin clusters.

REFERENCES

Fdez, E., and S. Hilfiker. 2006. Vesicle pools and synapsins: New insights into old enigmas. *Brain Cell Biology* 35:107–115.

Turner, K. M., R. D. Burgoyne, and A. Morgan. 1999. Protein phosphorylation and the regulation of synaptic membrane traffic. *Trends in Neurosciences* 22 (10): 459–464.

Wainer, G. 2002. CD++: a toolkit to develop DEVS models. *Software, Practice and Experience* 32 (3): 1261–1306.

Wainer, G., and N. Giambiasi. 2002. N-dimensional Cell-DEVS models. *Discrete Event Systems: Theory and Applications* 12 (1): 135–157.

Wainer, G., S. Jafer, B. Al-Aubidy, A. Dias, R. Bain, M. Dumontier, and J. Cheetham. 2007. Advanced DEVS models with application to biomedicine. *Artificial Intelligence, Simulation and Planning, Buenos Aires, Argentina*.

Wolfram, S. 2002. *A New Kind of Science*. Wolfram Media.

Zeigler, B., T. Kim, and H. Praehofer. 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press.

AUTHOR BIOGRAPHIES

RHYS GOLDSTEIN received a B.A.Sc. degree (2003) from the Engineering Physics Department at the University of British Columbia (Vancouver, BC, Canada). He then worked in the mining industry, developing data analysis software and leading geophysical surveys in various parts of the world. He is now pursuing a M.A.Sc. in Biomedical Engineering from the Department of Systems and Computer Engineering at Carleton University (Ottawa, ON, Canada). His email is `<rhys@sce.carleton.ca>`.

GABRIEL WAINER received the M.Sc. (1993) and Ph.D. degrees (1998, with highest honors) of the University of Buenos Aires, Argentina, and Université d'Aix-Marseille III, France. In July 2000, he joined the Department of Systems and Computer Engineering, Carleton University (Ottawa, ON, Canada), where he is now an Associate Professor. He has held positions at the Computer Science Department of the University of Buenos Aires, and visiting positions in numerous places, including the University of Arizona, LSIS (CNRS), University of Nice and INRIA Sophia-Antipolis (France). He is author of three books and over 160 research articles, and helped organizing over 70 conferences. He is Associate Editor of the Transactions of the SCS, and the International Journal of Simulation and Process Modeling. He was a member of the Board of Directors of the SCS, a chairman of the DEVS standardization study group (SISO). He is Director of the Ottawa Center of The McLeod Institute of Simulation Sciences and chair of the Ottawa M&SNet, and one of the investigators in the Carleton University Centre for Advanced Studies in Visualization and Simulation (V-Sim). His current research interests are related with modeling methodologies and tools, parallel/distributed simulation and real-time systems. His email and web addresses are `<gwainer@sce.carleton.ca>` and `<www.sce.carleton.ca/faculty/wainer>`.

JAMES J. CHEETHAM received a Ph.D. (1993) from the Department of Biochemistry at McMaster University, then moved to the Laboratory of Molecular and Cellular Neurobiology at The Rockefeller University in New York. There he was introduced to signal transduction in neurons, and focused mainly on the synapsins which continues to study. In 1996, after several years in Manhattan, he returned to Canada and accepted a position as Assistant Professor in the Biology Department at Carleton University. In 2000 he was promoted to Associate Professor. In 2001 he completed the Canadian Bioinformatics Workshops, given by the Canadian Genetic Disease Network, and has an increasing interest in bioinformatics research. His email is `<jcheetha@ccs.carleton.ca>` and his website is at `<http-server.carleton.ca/~jcheetha>`.

RODERICK S. BAIN received his Ph.D. in Chemical Engineering at the University of Wisconsin (1993). Subsequently he was a postdoctoral fellow at the University of Pennsylvania in the Department of Computer and Information Science. Currently he is a Research Associate in the Biology Department at Carleton University. His email is `<rod.bain@sympatico.ca>`.