A FRAMEWORK FOR COMPOSABLE CELLULAR AUTOMATA DEVS MODELING, SIMULATION, AND VISUALIZATION

Chao Zhang
Hessam S. Sarjoughian

Arizona Center for Integrative Modeling and Simulation
Schl. of Computing, Informatics & Decision Sys. Engr.
Arizona State University, Tempe, AZ, USA, 85261-8809
{czhang72, sarjoughian}@asu.edu

Moon G. Seok

School of Computer Science and Engineering
Nanyang Technical University
Singapore, 639798
moongi.seok@ntu.edu.sg

ABSTRACT

The concept of cellular automata is one of the cornerstones of scientific and engineering pursuit with wide applicability in various domains. The rise of system complexity points to developing frameworks that can aid in developing composable models at multiple spatiotemporal levels. Toward this goal, an input/output modular discrete-event cellular automata framework based on the Discrete Event System Specification (DEVS) is proposed. It is adapted from a Composable Cellular Automata (CCA) formalism that is defined in terms of a multicomponent discrete time system specification. The resulting Composable Cellular Automata DEVS (CCA-DEVS) framework is built using the Cellular Automata DEVS that is a part of the DEVS-Suite simulator. The approach and realization of the proposed CCA-DEVS framework are detailed. The framework is used to develop an example model composed of CCA-DEVS agent and partial difference equation models.

Keywords: Composable Cellular Automata, Composability, DEVS, DEVS-Suite Simulator

1 INTRODUCTION

It is common to study different kinds of natural and built systems solely using the cellular automata (CA) models. This is, in part, due to the inclusion of geometric abstraction which in turn results in cellular automata being more expressive than other kinds of automata (Toffoli and Margolus 1990). A cellular automaton has a set of individual cells that have a set of common properties and synchronously change their dynamics with respect to one another using a set of shared rules (Von Neumann 1966). The cells change their states under a uniform or non-uniform local neighborhood configuration. It is important to observe that the use of space and time together in modeling dynamical systems has a unique benefit in the same way Partial Differential Equations are more expressive than Ordinary Differential Equation. This view is obvious in the natural and built worlds and presents cellular automata to be well-suited for understanding and developing hybrid systems among others (Mayer and Sarjoughian 2016).

Cellular automata are commonly used for single abstraction levels of a system. For example, a CA specification for an airport roadway system can be defined to have seconds and centimeter scales. The movement of people in the airport terminals connected by the roadways can have minute and meter scales. Thus, an airport can be modeled at multiple abstraction levels (temporal and spatial resolutions), one for the movement of people and another for the movement of vehicles. It is useful to have these abstractions to be on the one hand independent of one another, and on the other hand to be combined.

The cellular automata modeling approach, as in many others, allows abstracting systems as self-contained – i.e., the input, output, state, and function of a system are formulated to represent its inner structure and behavior. The input and output for cellular automaton are fundamentally are those that belong to its cells that are connected among the parts (neighborhood). The behavior is defined as the collection of the state of all cells computed using a time-based function applied to each cell's neighborhood. Therefore, cellular automata are attractive for describing phenomena that intrinsically have intertwined space and time aspects.

To study complex systems, such as an airport, it is useful to compose multiple cellular automata, each having its own space, neighborhood, state, and dynamics. A concept for composing CA is for each CA to be a modular in terms of having exogeneous input and output (Mayer and Sarjoughian 2009). A discrete-time formulation of this concept based on system theory called Composable Cellular Automata was introduced based on the multicomponent system specification. In this CCA Discrete Time Multicomponent (CCA-DTM), states of all cells are uniformly computed using a discrete time base. The behavior of every influenced cell can change due to its own state as well as the state of one or more influencing neighbors. The (partial) state of every influencing neighbor is its output. The collection of the neighbors' output constitutes the influenced endogenous input. In the CCA-DTM, exogenous I/O specification is added as the means to separate the inside of the CA specification from other automata (cellular or otherwise). As a standalone model, CCA-DTM can be composed with any other discrete-time I/O system specification (Zeigler, Kim, and Praehofer 2000). In other words, the CCA-DTM formalism is developed such that it can be coupled with I/O modular CA and non-CA discrete-time models.

1.1 Motivation

The composition of models is key to developing simulations for many domains. The aim of this research has been to support the composition of models that have strong input/output modularity (i.e., communication between different models is only via input and output ports). Toward this goal, the CCA-DTM concept was purposed for composing any discrete spatiotemporal I/O multicomponent system specification with other I/O specified models. In this paper, the CCA Discrete Time Multicomponent formalism (Mayer and Sarjoughian 2009) is adapted to allow both the cells and the CA to be I/O modular and have a continuous-time base. Specifically, a CCA-DEVS formalism is introduced and a framework for it is developed by extending the CA-DEVS (Zhang and Sarjoughian 2017). This has resulted in extending the DEVS-Suite simulator (ACIMS 2020).

The remainder of this paper continues with Section 2 that provides a review of the DEVS and CCA-DTM formalisms with the CA-DEVS framework. In Section 3, the DEVS counterpart of the CCA-DTM formalism is described followed by a description of the CCA-DEVS framework in Section 4. An agent-diffusion CCA-DEVS model developed in the CCA-DEVS framework is described in Section 5. A comparison of the CCA-DEVS modeling with those of the Cell-DEVS and Complex Automata is provided in Section 6. Some observations are discussed in Section 7. The conclusion and future work is included in Section 8.

2 BACKGROUND

The basic idea of the CCA-DEVS is to apply the discrete event manner to the regular discrete time CCA. CCA-DEVS is a combination of the existing works of DEVS and CCA. In the meanwhile, CA-DEVS is a platform built for developing cellular automata model in DEVS format. If CCA is extended to CCA-DEVS, the CA-DEVS platform can also be extended to support CCA. Within the CCA-DEVS platform, CCA can either have a discrete time manner or in a discrete event simulation. To better understand the definition of DEVS and CCA, as well as the design of the CA-DEVS platform, the backgrounds will be introduced here.

2.1 Multicomponent and coupled systems

The multicomponent system specification (Zeigler, Kim, and Praehofer 2000) is used to define the Multicomponent Discrete Time CCA formalism (Mayer and Sarjoughian 2009). The components belonging to the multicomponent system are void of inputs and outputs. Each component has a state set and a state transition function. Any interaction between two given components is defined using the influencer-influencee concept. A component is influenced by its influencers and may influence other components through its state transition function - i.e., components do not have I/O modularity. All state transitions are driven by a discrete-time clock. A multicomponent system specification has a flat structure. It has input and output. They are intended to be used to couple the model with the outside world (other models). The components have external input and output since the multicomponent system specification has input and output. A multicomponent system specification is defined as $MC = \langle T, X, \Omega, Y, D, \{M_d\}\rangle$ where T is a time base, X is a set of input values, and Y is a set of output values. Each component $d (d \in D)$ is defined as $M_d = \langle Q_d, E_d, I_d, \Delta_d, \Lambda_d\rangle$. The Q_d is a set of states of component d. The $I_d \subseteq D$ is a set of components influencing d (called influencers). The $E_d \subseteq D$ is a set of components influenced by d (called influencees). Each component has a state transition function $\Delta_d : \times_{i\in I_d} Q_i \times \Omega \rightarrow \times_{j\in E_d} Q_j$ and an output function $\Lambda_d : \times_{i\in I_d} Q_i \times \Omega \rightarrow Y$. Each local state transition function takes the influencers' state and maps them into the influencees' new states. The local output function of every influencer uses its state and computes an output. These component outputs represent the output of the whole system. The multicomponent system specification is specialized to support differential equations, discrete-time, and discrete-event system specifications. The components of the Multicomponent Discrete Time System Specification can be spatially bounded to become cellular automata.

The coupled system specification consists of I/O modular components with a strict hierarchical tree structure. Each component receives inputs from its influencers through unidirectional I/O couplings. The inputs and the current state of each component are used to compute the component's next state. For some coupled system specifications, there exists the closure under coupling property which means the resultant of the composition of the components is itself an I/O modular component. The DEVS formalism, as a variant of the coupled system specification, has this property with a continuous time base and no or a finite number of state transitions for any finite time period.

2.2 Multicomponent Composable Cellular Automata

To compose a cellular automata model with other models, the Composable Cellular Automata Discrete Time Multicomponent (CCA-DTM) specification (Mayer and Sarjoughian 2009) was introduced. This Composable Cellular Automata specification is defined as a network of a two/three-dimensional multicomponent discrete-time system. Every cell is defined using the Discrete Time System Specification (DTSS) (Zeigler, Kim, and Praehofer 2000). It is a standalone model with the ability to be composed with other CCA-DTM or I/O modular models. The CCA can communicate with other models via I/O and external mapping functions. Communication within the CCA is through the influencer-influencee relationship and computed with local state transition and output functions defined for multicomponent system specification. Thus, the cells in an CCA-DTM may have input events from either its neighbors (i.e., influencers) or from external models coupled with the CCA-DTM model. The output from the cell will be used by its neighbors (i.e., influencee) or directly to the outside of the network through the I/O ports to other sub-models. The CCA-DTM can be composed using couplings with other I/O modular models. It can also be combined with non-modular I/O models through direct use of input/output, for example as in the multicomponent system specification. This specification is based on the discrete time system formalism. All cells in the CCA are updating simultaneously based on a common time interval. The structure and behavior of the cells, with respect to the inside of the CCA, are identical to those defined for the MDT Specification.

The communication between CCA-DTM and the external system is archived through two mapping functions. The output from each cell will be mapped to the CCA-DTM and then mapped to the input of the external system. Also, the input from the external system will be mapped to the entire network and then to the individual cells. The CCA-DTM does not include the specification for these external mapping functions, but they are necessary for the CCA-DTM to be used in hybrid models (Mayer and Sarjoughian 2009). The mapping functions are used to couple any number of models. Given differences in the structure, spatial, and temporal of any two models, the external I/O mapping functions may be partial. To compose CCA-DTM with other kinds of model types, the polyformalism model composition approach defined as the Knowledge Interchange Broker (KIB) (Sarjoughian 2006) is used.

Since cellular automata is a network of individual cells, there is a possibility that two or more cells may affect one another. Under this condition, the states of the cells must be independent of another. Given an influencee cell, its state cannot be used at the same time it depends on the state of the cell it may influence it. This constraint eliminates the possibility of a delayless feedback loop and ensures the cellular automata to be causal (i.e., the next states of two cells can depend only on each other's prior state or output).

A realization of the CCA-DTM is developed for GRASS, which is a standalone system. Since GRASS has no direct representation for time and lacks I/O modularity, it is wrapped in a parallel DEVS atomic model. The atomic model is restricted to have a discrete time base. The cells' internal state transitions are synchronized to execute according to the parallel model simulator clock. Every cell in the CCA-DTM simultaneously executes with all other cells in a step-wise fashion.

2.3 Discrete Event System Specification

Discrete systems with hierarchical structures and time-based behaviors can be specified using the Parallel DEVS modeling formalism (Zeigler, Kim, and Praehofer 2000). A system can be modeled using atomic and hierarchical coupled models. Atomic models can have arbitrary behaviors. All the interactions between model components are strictly through I/O ports with unidirectional couplings. The atomic and coupled models are indistinguishable in terms of their I/O. Models execute concurrently in accordance with the Parallel DEVS abstract simulator protocol.

The behavior of the DEVS system is governed by state changes due to internal and external events defined in atomic models and couplings defined in coupled models. An atomic model has internal state changes. When the duration assigned for a state change is finite, output events can be generated followed and the change of state. The state changes can also happen due to input events. The atomic model is defined as the mathematical structure $\langle X^b, S, Y^b, \delta_{int}, \delta_{ext}, \delta_{conf}, \lambda, ta \rangle$ where X and Y are the input and output sets, S is the state set, $\delta_{int} : S \to S$ is the internal transition function, $\delta_{ext} : Q \times X \to S$ is the external transition function with Q being the set of total states ($Q = \{(s,e), s \in S, 0 \leq e \leq ta(s)\}$), $\delta_{conf} : Q \times X^b \to S$ is the confluent transition function for resolving simultaneous external and internal events, $\lambda : S \to Y^b$ is the output function, and $ta : S \to R_0^{+\infty}$ is the time advance function.

The coupled model is also defined as the mathematical structure $\langle X^b, Y^b, D, \{M_d, d \in D\}, EIC, IC, EOC \rangle$. It has input and output sets which are X^b and Y^b, an index set D representing the names of its atomic and coupled models, a set of atomic and coupled models M_d, a set of external input couplings EIC for receiving inputs from X^b, a set of external output couplings EOC for sending outputs to Y^b, and a set of internal couplings IC for atomic and coupled models within M_d to send/receive events. Every model has a strict tree hierarchy where its leaf nodes are atomic models and all other nodes are coupled models. The edges in the tree are couplings amongst input and output ports.

2.4 Cellular Automata DEVS Framework

The CA-DEVS framework is developed for Cellular Automata models that conform to the Parallel DEVS formalism (Zhang and Sarjoughian 2017). In this framework, atomic and coupled DEVS models with assigned spatial configurations are cells and cellular automata, respectively. Cells execute concurrently with $[0, \infty]$ timing. The CA-DEVS complements the DEVS-Suite simulator component view with a cellular grid view. Dynamics of any CA or any contiguous part of it can be dynamically visualized in a two-dimensional space. Other views and control features of the DEVS-Suite simulator have been extended to support CA-DEVS. All or any subset of the input, output, and state time trajectories as linear and superdense time of any number of cells can be visualized independently of one another. This is achieved by extending the DEVS-Suite simulator's Timeview. State change visualization in two-dimensional space is supported with playback. Input, output, and state of any number of cells can be plotted and/or tabulated. Each cell's dynamics can be tracked at its own start time.

Cellular automata models usually stepped in a discrete time manner where all the cells in CA evolve under its internal state change simultaneously. For simulation speed-up, the discrete event base (i.e., a continuous time base having a finite number of time instances) is used. As a result, quiescent cells and their neighbors do not need to undergo state change. I/O modularity and zero-time state transition remove the synchronization restriction provided no cell can have feedback to itself DEVS (Parallel DEVS legitimacy property is satisfied (Zeigler, Kim, and Praehofer 2000)). The cells in the CA-DEVS framework will only interact via input and output, and no direct visibility or manipulation of the state by any cells is allowed. Although I/O coupling leads to high computational cost (slows simulation speed), the modularity allows every cell's internal and external behavior to be totally transparent. Consequently, the development and debugging of cellular automat models benefit from combined animation and arbitrary amounts of details on each cell's internal and I/O dynamics. The CA-DEVS framework for modeling, simulation, and visualization makes it desirable to be extended for CCA-DEVS.

3 CCA-DEVS MODELING

The idea of the CCA-DEVS is to adapt the CCA-DTM to be internally I/O modular according to the Parallel DEVS formalism. This is achieved by replacing models of cells from discrete time to discrete event system specification and multicomponent to a network of system specification (coupled systems). The result is the DEVS atomic models describe cells are geometrically arranged and coupled together to represent composable cellular automata models. In this approach, the cells and the CCA are all I/O modular, thus readily supporting composing it CCA-DEVS with any other I/O modular system specification including Parallel DEVS atomic and coupled models.

The individual cells in a CCA-DEVS model have input and output ports connected to each other and also to other modules. As a result, hierarchical cellular automata models can be specified. Furthermore, the discrete-event cellular automaton and the discrete-event cells have a common continuous time base. The external mapping functions for CCA-DEVS conform to the atomic and coupled DEVS system specifications.

3.1 Approach

The DEVS formalism is modularized by using ports of I/O for communication, while the CCA-DTM system only has external ports for coupling with other models. The hierarchical structure of DEVS is organized from the atomic model to the coupled model, while the CCA-DTM is a standalone I/O system without hierarchical structure. The cells in CCA-DTM have no input and output interfaces. When combining a CCA with other CCA or non-CCA models, time synchronization for all the sub-systems (CCA or non-CCA) is based on the

same time factor with a natural number multiplier in the discrete time manner. The new way of modeling CCA using DEVS is changing the system timing to a discrete event base, such that each component haves its own timing. This allows coupling CCA model with other types of I/O modular models that have continuous or discrete event time bases.

In the discrete-event abstraction, a finite number of events may occur during a bounded time period. One of the differences between discrete event and discrete time is that the time interval in discrete event is not fixed. The time step in discrete time abstraction is a natural number and the time base in discrete event is a non-negative real number. The time in the modeling and simulation domain is mathematically defined to be either discrete or continuous. For CCA-DTM, time is defined as $T = \{h_m | 0 \leq m \leq n\}$. This is an ordered, finite set of time intervals, h_m, (i.e. $\{h_0, h_1, h_2, \ldots, h_n\}$) such that $m, n \in \{(N \cup \{0\}) - \{\infty\}\}$. In the CCA-DEVS specification, time is a part of the state and defined as $\sigma \in R^{+}_{0,\infty}$.

There is no predefined time list (scheduling) for discrete event specification, neither for the cells nor for the whole CCA. Each component may change its state through the input it receives either via its influencing neighbors or from outside the model. The second difference between discrete event CCA with discrete time CCA is handling of state transition. In discrete time CCA, the state transition function of M_{ijk} is defined as $\delta_{ijk} : Q_{ijk} \times X_{ijk} \rightarrow Q_{ijk}$, that its next state $q' \in Q_{ijk}$ at time t_{r+1} is computed using the current state, $q \in Q_{ijk}$ at time t_r and the set of inputs, X_{ijk} at time t_r. In the discrete event CCA, the state transition function is using the external, internal, and confluent transition functions defined for the atomic DEVS model. For the CCA-DEVS, the network N is specified as:

$$N = \langle X_N, Y_N, D, \{M_{ijk}\}, F, IC \rangle, \text{ where}$$

$X_N = \{\bar{X}_{ijk} | (i,j,k) \in D\} \vee \emptyset$ is the external input mapped to the network, N,

$Y_N = \{\bar{Y}_{ijk} | (i,j,k) \in D\} \vee \emptyset$ is the external output for $\{M_{ijk}\}$ from the network, N,

$F = f_{out}, f_{in}$ is the set of internal I/O mapping functions between the network and its cell components $\{M_{ijk}\}$ where $f_{out} : \bigcup_{(i,j,k)\in D} \bar{Y}_{ijk} \mapsto Y_N$ and $f_{in} : X_N \mapsto \bigcup_{(i,j,k)\in D} \bar{X}_{ijk}$,

$D = \{(i,j,k) | a \leq i \leq b, c \leq j \leq d, e \leq k \leq f\}$ is the index set where $a, b, c, d, e, f \in \mathbb{Z}$,

$IC \subset \bigcup_{(i,j,k)\in D} \dot{Y}_{ijk} \mapsto \bigcup_{(i,j,k)\in D} \dot{X}_{ijk}$ is the internal coupling between cells.

The component M_{ijk} has the same specification as an atomic DEVS model with the constraints that the cells are uniformly coupled according to a cellular spatial specification $M_{ijk} = \langle X_{ijk}, Y_{ijk}, S_{ijk}, \delta_{ijk(int)}, \delta_{ijk(ext)}, \delta_{ijk(conf)}, \lambda_{ijk}, ta \rangle$. The set $X_{ijk} = \dot{X}_{ijk} \cup \bar{X}_{ijk}$ is the input where $\dot{X}_{ijk} = \bigcup_{l \in I_{ijk}} (\dot{Y}_l)$ is the input for M_{ijk} originating from the set of the output of its influencers, $M_l | \forall_l \in I_{ijk}$, and $\bar{X}_{ijk} \subseteq X_N$ is the input originating from outside the network, and N, which is mapped to M_{ijk}, and $\bar{X}_{ijk}$ might be $\emptyset$. The set $Y_{ijk} = \dot{Y}_{ijk} \cup \bar{Y}_{ijk}$ is the output from M_{ijk}, where $\dot{Y}_{ijk}$ is the output from M_{ijk} that acts as input to the cells that M_{ijk} influences and $\bar{Y}_{ijk} \subseteq Y_N$ is the output from M_{ijk} that contributes to the network output, Y_N, and $\bar{Y}_{ijk}$ might be $\emptyset$. The internal couplings $I_{ijk} = \bigcup_{a=i-1}^{i+1} \bigcup_{b=j-1}^{j+1} \bigcup_{c=k-1}^{k+1} (a,b,c)$ is for a three-dimensional network. It defines the neighbors of cells (e.g., Moore neighborhood structure). The remaining elements of M_{ijk} are as defined in the parallel atomic DEVS model.

The network abstraction, in part, has the internal spatial structure and how it is exposed to the outside. The automaton abstraction defines the individual cell components. Each cell's influencers and influences are predefined. The inputs of each automaton are either from the inside of the network (i.e., its neighbors) or from the outside of the network. The generated output from cells is sent to its connected influencees and to the outside of the network per network's external I/O mapping function. The internal state of the cell can be exposed to the outside as output.

The external I/O mapping functions for network N defines its coupling to others models that may or may not be a CCA. The external system Θ and the set of mapping functions $G = \{g_{out}, g_{in}\}$ define mapping of external data from N to N and vice versa. The $g_{out} : Y_N \mapsto \Theta_{input}$ function defines the output from the CA network N, mapped to the input of Θ. The $g_{in} : \Theta_{output} \mapsto X_N$ function defines the input from Θ mapped to the network. The mappings from the external system, Θ to the inside cells, M_{ijk}, and backwards, are defined as mapping functions: $g_{out} \circ f_{out} : \bigcup_{(i,j,k)\in D} \overline{Y}_{ijk} \mapsto \Theta_{input}$ and $f_{in} \circ g_{in} : \Theta_{output} \mapsto \bigcup_{(i,j,k)\in D} \overline{X}_{ijk}$. As g_{out}, g_{in} and G are not within the CCA, they are not inside the CCA specification, nor the CCA-DEVS specification. Different CCAs can be coupled together to form a whole CCA-DEVS model. These mapping functions are defined as a simple interaction model using atomic DEVS models. A CCA-DEVS model can be coupled to a non-CA model using an interaction model too. The interaction model can be specified to have data transformations, timing, and control as needed for the application domain of interest using the Knowledge Interchange Broker approach (Sarjoughian 2006).

4 CCA-DEVS IMPLEMENTATION

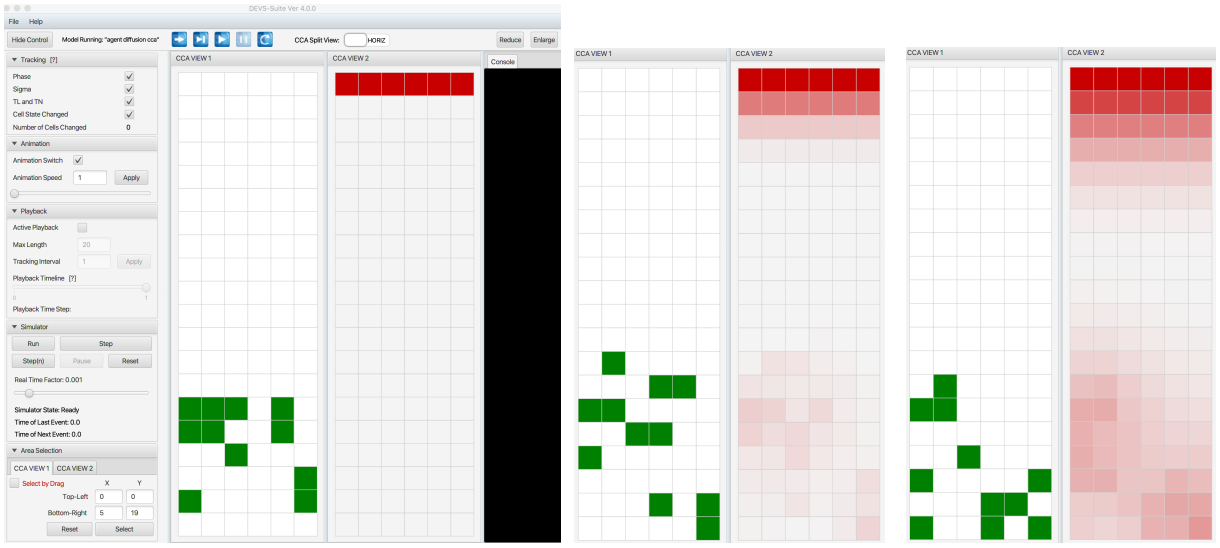
The DEVS-Suite simulator environment supports component-based parallel DEVS model (Kim, Sarjoughian, and Elamvazhuthi 2009), as well as cellular automata-based DEVS model (Zhang and Sarjoughian 2017). CA-DEVS models can be simulated as coupled DEVS component models with the same level of dynamic visualization (e.g., animation of message passing and time-based trajectories) as well as run-time 2D cellular automata animation in a grid space and other features such as playback (Zhang and Sarjoughian 2017). The speed of the animation and changing display area can be modified. These and other features are supported in CCA-DEVS. The extension of the simulator supports having separate 2D animation grid space frames. Every composed CCA model, rendered in its own grid space, can be dynamically viewed together. Figure 1 shows a sample model composed of two independent CCA-DEVS models.

Since every CCA-DEVS is a discrete event model, animation of the composed CAs is asynchronous. Cells can have intermediate states due to zero time advance and or confluent state transition function. These intermediate states are displayed during 2D simulations. This level of comprehensive visualization enriches understanding of CCA dynamics and emergence. For time trajectories, simulation can be configured to exclude state changes with zero time advance. CCA-DEVS models can be easily constrained to have a discrete time characteristic by restricting time assigned to all state changes to be $\mathbb{N} - \{\infty\}$. Many existing cellular automata models and agent-based models have a discrete time base which allows their reuse.

There exists two I/O coupling methods for CCA-DEVS models. In one, all cells of one model are connected to all the cells of another. In another, I/O couplings are at the CCA level – i.e., only external messages received from the outside are propagated to the cells inside the CCA. That is, only specified cells can be activated by some messages with all other cells ignoring the messages. When two CCA-DEVS models are not on the same spatial scale or on the same scale but with different size, the models cannot be coupled. For this class of CCA-DEVS, the Knowledge Interchange Broker modeling approach can be used. Besides, it is possible to compose more than two models that can be simulated and visualized together.

5 EXAMPLE CCA-DEVS MODEL

To show the use of the CCA-DEVS, two CCA-DEVS model types are developed and composed. One type is a PDE diffusion and another type is agent-based. These two models have the same space dimension. The CCAs are coupling using the one-to-one coupling method. Examples where PDE and agent-based model can have different spatial scales have been developed, but not included in this paper. The PDE is specified using the finite difference scheme (Recktenwald 2004). This scheme is a numerical method where the partial differential equations are discretized in space (divided into a regular grid) and time (discrete time step). Although stability is a key part of development of PDE models, for the purpose of this work, it is



(a) $t = 0$ (b) $t = 5$ (c) $t = 20$

Figure 1: Runtime screenshots of a simulated composite CCA-DEVS diffusion and agent model.

assumed appropriate techniques (e.g., choice of solvers) for discretization is available (Cellier and Kofman 2006). It is also assumed the actions of the agents on the individual and collection of the cells of the discrete PDE model do not undermine some desired precise and accurate simulated dynamics.

To solve the PDE of diffusion, a popular finite difference method can be used. The finite difference scheme is a numerical method that discretizes partial differential equations on both space and time. Space is approximately discretized on a regular grid of point and the state variable is evolved over the discrete time step. The density of the diffusion material is defined for a uniform grid with locations (x,y) and time instances $t_n = t_0 + n\Delta t$. A simple way to discretrize the PDE is to use the explicit FTCS (Forward-Time Central-Space) method (Recktenwald 2004) with a limit on time discretization satisfying some stability condition (e.g., $\Delta t \leq \frac{\Delta x^2}{4D}$ for $\Delta x = \Delta y$ with D as a diffusion coefficient). It is noted that cellular automata is defined as an alternative to differential equations for physics modeling (Toffoli and Margolus 1990). In other words, CCA-DEVS can be used to specify a Partial Difference Equation Diffusion model.

A class of agent models supports simulating the movement of agents in a cellular space. Agents can move in combinations of continuous and discrete space and time. The CCA-DEVS supports regular grid space with continuous and/or discrete time base. In an agent CCA model, each CA grid cell is assigned an atomic model (i.e., an agent). An agent's movement is defined by switching its state from not empty to empty. When the discrete event time base is used, an agent can have different movement speeds while some other agents do not move or have a role in the movement of other agents.

After the CCA diffusion model and the CCA agent moving model are created using the CCA-DEVS capability, the models can be added to the DEVS-Suite simulator. Since the diffusion model has a discrete time base with the FTCS solver and the agent moving model has a continuous time base, the combination model will update with minimal step size, but not all the atomic DEVS model components need to be executed. The combined model is a simplified cancer disease model (Chang, Cavnar, Takayama, Luker, and Linderman 2015). The model is for the CXCL12 chemotaxis, and the chemokine protein diffusion in human tissue. The diffusion begins at the high density at one end to the low density at the other end. The agents in the other model are the CXCL12-Cell which can secrete the CXCL12 protein into human tissue. The CXCL12-Cell agents move randomly while releasing some amount of CXCL12 protein. The coupling between these two

CCA models is uni-directional. It is from the CXCL12 protein agents to the CXCL12 chemotaxis diffusion process. The benefit of separating the diffusion and agent models is important for tracking and inspecting the dynamics of the cells selectively and individually as well as analyzing the influence of protein movement in human tissue. Example 2D displays of the states of these models at (a) $t = 0$, (b) $t = 5$, and (c) $t = 20$ time instances are shown in figure 1.

6 RELATED WORKS

Other approaches have been developed that share the basic needs described above. The approaches that benefit from component concept are Cell-DEVS (Wainer 2017) and Complex Automata (Hoekstra, Kroc, and Sloot 2010). The former extends the DEVS formalism. The latter, recently called multiscale modeling and simulation framework (MMSF), views a CA to be composed of multiple CAs. Next, these approaches are reviewed and then compared with the CCA-DEVS approach.

Cell-DEVS: This approach incorporates cellular automata into the DEVS formalism. The Cell-DEVS is based on the idea that a cell is a building block (i.e., a component) in the way an atomic DEVS model is with the addition of having a designated spatial representation. These spatial, time-based, I/O modular cells are coupled to represent cellular automata models. A cell is formalized as $TDC = \langle X, Y, S, N, d, type, \tau, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$ where X and Y are the sets for external inputs and outputs with S representing the state set. The set N is for the inputs from the neighbor cells which can be external Cell-DEVS or external DEVS models. Each cell computes its next state by using its current state and the inputs from its neighbors using the local computing function τ. The delay assigned to all the cells is either *inertial* or *transport* type with a duration d. The transport refers to the delay period for all the activities of the model with variable commuting time, while with inertial delays, any scheduled events will be removed if the computed state is different from the future state. The remaining elements $\delta_{int}, \delta_{ext}, \delta_{conf}, \lambda$ and ta are the same as those defined for the atomic model.

The Cell-DEVS coupled model is specified as $GCC = \langle X, Y, I, X_{list}, Y_{list}, \eta, \{t_1, \cdots, t_n\}, N, C, B, Z, select \rangle$ where X and Y are the sets of external input and output events. The X_{list} and Y_{list} are the input and output coupling lists. The I is the interface of the model and η is the neighborhood size. The set $\{t_1, \cdots, t_n\}$ represents the number of cells in each dimension. The set N represents the neighborhood and C is the cell space. The set B represents border cells with Z being a mapping function for coupling the cells. The function select serves as a tie-breaking as defined for coupled DEVS models. Of interest are the sets X, Y, and N defined for Cell-DEVS atomic model and X, Y, and N defined for the Cell-DEVS coupled model in view of the sets X_N, Y_N, F, IC defined for the CCA-DEVS specification. Other DEVS-based approaches have been developed for combining agents and CA models (e.g., (Hu, Muzy, and Ntaimo 2005)). In such works, the DEVS coupling is used to combine models statically or dynamically.

Complex Automata: The Complex Automata (CxA) (Chopard, Borgdorff, and Hoekstra 2014) approach and framework are proposed for multiscale and multi-science phenomena modeling and simulation. The framework can be used to define a multiscale model as a collection of single scale sub-models. It uses the idea of decomposing a CA into multiple single scale CA models, each having its own temporal and spatial scales. Each CA sub-model is a component that may interact with other CA sub-models. A single scale CA is defined as $C = \{A(\Delta x, L, \Delta t, T), \mathbb{F}, \Phi, f_{init} \in \mathbb{F}, u, O\}$. The A is a domain consisting of spatial cells of size Δx and forming an area of size L. The quantity Δt represents a time step, T represents the end of a simulated time period, and $T/\Delta t$ is the time scale. The state of the CA is defined by an element of $\mathbb{F}$ with the update rule $\Phi : \mathbb{F} \rightarrow \mathbb{F}$, both of which depend on discretized Δx and Δt. The update rule is constrained by propagation, collision, and boundary such that it can be decomposed to propagation (sending data to neighbors), collision (updating the state of cells), and boundary condition. The initial condition f_{init} is an element of the space of states $\mathbb{F}$. The field u is a collection of external information that is exchanged between

CA and its outside environment at each iteration. The functional O represents the observable quantity of interest (output).

A Complex Automata is a collection of interacting CA models. It can be viewed as a graph $X = (V, E)$ where V is a set of vertexes which are a collection of CA: $C_i = \{A_i(\Delta x_i, L_i, \Delta t_i, T_i), \mathbb{F}_i, \Phi_i, f_{init,i} \in \mathbb{F}_i, u_i, O_i\}$, and each edge $E_{i,j}$ is a coupling between two subsystem C_i and C_j. This form of coupling allows any two CAs not to occupy a contiguous space. To support this kind of coupled CA, the concept of scale bridging is proposed. Models that have different spatial and temporal scales can be coupled. For scale resolution, a variety of methods (e.g., sampling, projection, splitting, lifting/upscaling, homogenization/coarse gaining, refinement, micro-macro coupling, constitutive models, boundary methods) are proposed. These can be used to bridge disparities among sub-models. For example, in some multi-scale applications, cell movement at the micro-scale resolution is coupled to finite-element fluid mechanics at macro-scale resolution. The plain, filter, and mapper bridging methods must be found and defined in order to couple sub-models.

Models with different spatial scales can be composed using map functions. The scale bridging methods are represented by the directional edge in the scale separation map. It defines the computational and scale requirements for sub-models and the bridging components, each of which has predefined output and input ports. Data are exchanged and communicated via these ports. The data exchange can be supplemented by data mappings to match scale differences. The CxA framework supports multiscale modeling language, sub-model execution loop, and coupling templates. The specifics of these conduits should be formulated by modelers. Every cellular automaton is abstracted to a component to be composed with other cellular automata. Each sub-model is graphically represented as a Lego with connections shown as lines (same as component visualization in the DEVS-Suite simulator).

7 OBSERVATIONS

Considering the Cell-DEVS and CCA-DEVS approaches, they both support the composition of spatial atomic and coupled models according to the DEVS formalism. Both use discrete event time base, allowing state updates triggered by input events or passage of time. They support coupling I/O modular CA with other CA or non-CA model. In other words, a system that can be modeled in one can also be modeled in the other. However, there are some important differences between Cell-DEVS and CCA-DEVS from the composability vantage point. One difference is that the composed models in CCA-DEVS are strictly distinct from one another. Coupling two CCAs is via mapping functions with couplings instead of only couplings. This means an external coupling of a CCA with another model is not included in the model that is being composed of some other model. Another difference is that Cell-DEVS uses local computing function which will have access to the neighbor cell's state information. Cell-DEVS explicitly defines boundary conditions. In CCA-DEVS, the boundary condition is included in the external I/O mapping function. Consequently, the cells may not form a boundary at the edge of CCA-DEVS.

The CxA is aiming to build a universal solution for combining separate sub-models with different temporal or spatial scale. Its framework supports sub-models in different programming languages to be executed and integrated among different computing platforms. The CCA-DEVS supports multiscale modeling using interaction modeling. Inside CxA, the sub-model is wrapped as an individual module, and then the interaction between modules is though boundary condition and collision operation. The time in CxA is in discrete steps, which is implicitly treated inside the update function. Of importance is that CCA-DEVS supports visualizing run-time execution of composed models side-by-side. During a simulation, the effect of changes (shown as different colors) in one model can be observed in the other. Furthermore, time-based state, input, and output trajectories of any cell of a CCA can be independently tracked with arbitrary starting time. The stored data can be post-processed and displayed using visualization tools. All CCA-DEVS models can be simulated at a significantly faster speed without data storage and run-time visualization.

2D Cell Space		Run-time visualization	Logical time	Physical Time (seconds)		Performance Degradation CA→Coupled CCA
$x \times y$[# of cells] for CA	$x\times$ y[# of cells] for Coupled CCA		Duration	CA Diffusion	Coupled CCA Diffusion	
$[30 \times 20]$	$[25 \times 20]$ + $[5 \times 20]$	No	100	1.04	1.23	18%
		Yes	100	1.90	2.39	26%
		No	1000	9.52	11.94	25%
		Yes	1000	18.21	24.09	32%
$[150 \times 100]$	$[125 \times 100]$ + $[25 \times 100]$	No	100	19.10	22.82	19%
		No	1000	202.28	282.04	39%

Table 1: A sample comparison of the CA-DEVS and coupled CCA-DEVS diffusion models

For illustration, a small set of simulation experiments are developed to highlight the computational cost of coupling CCAs compared with having one CA model. Two simple CA diffusion models with dimensions $x_1 = 30 \times y_1 = 20$ and $x_2 = 150 \times y_2 = 100$ are developed in the DEVS-Suite simulator v. 5.0. Two coupled CCA models, each having two CCA models are constructed from the CA models. One coupled CCA has CCA models with $x_{11} = 25 \times y_{11} = 20$ and $x_{12} = 5 \times y_{12} = 20$. The other coupled CCAs has CCA models with $x_{21} = 125 \times y_{21} = 100$ and $x_{22} = 25 \times y_{22} = 100$. For each simulation experiment, the coupled CCAs are spatially adjacent to one another. Interaction models are defined for the coupled CCAs are $x' = 1 \times y' = 20$ and $x'' = 1 \times y'' = 100$. This mapping is defined in an atomic model that accounts for the coupled CCA external inputs and outputs ($\overline{X}$ and $\overline{Y}$ defined for CCA in Section 3.1). For simplicity, all the cells $(x_i, y_j, i, j \in \{2, \cdots, n\})$ in the coupled CCA models have identical size. The y dimensions of the coupled CCAs are equal. The interaction for each coupled CCA is defined as an atomic model. It receives input events from the CCAs' boundary cells which are subsequently processed and sent to the CCAs' boundary cells. The execution times for the coupled CCA models are higher than those of the CA models due to the interaction models (see Table 1). The sample performance data (measured in physical time) shown in Table 1 are the averages of five simulation runs. The physical time duration for simulating the above diffusion CA models with $x = 150, y = 100$ dimensions for 100 logical time units using the Cell-DEVS framework are 321 seconds. This execution time includes writing an ASCII log file. The CxA framework is not available to be evaluated for computational efficiency. A desktop computer with Intel i7 2.60 GHz CPU, 64 GB RAM, and Windows 10 OS is used to run the simulation experiments using the DEVS-Suite and Cell-DEVS simulators.

8 CONCLUSIONS

A framework based on the composable cellular automata and parallel DEVS modeling approaches is proposed. This research has focused on model composability at the I/O level both within and among individual cellular automaton DEVS models. As part of the DEVS-Suite simulator, it supports developing composition and execution of CCA models that afford complementary run-time textual and 2D CCA displays with time-based input, state, and output trajectories. The design and implementation of the framework enable configuration and dynamic 2D displays with run-time execution and visualization control mechanisms. An example coupled diffusion and agent models is developed to demonstrate the importance and the use of discrete-event cellular automata models. This approach for composing cellular automata models can help to understand better cell-to-cell as well as CCA-to-CCA interactions. Consequently, alternative or better designs for compositions of multiple cellular automata can be developed. Future research includes empowering composability using the Knowledge Interchange Broker approach, where interactions amongst composed CCA can be defined in a standalone fashion. The interactions between CCAs can be modulated in terms of data transformations, control schemes, distributed execution, and time granularity. Another area of research is to address performance degradation due to the exponential increase in the number of I/O couplings as the scale of CCAs increases.

REFERENCES

ACIMS 2020. "DEVS-Suite Simulator". https://acims.asu.edu/software/devs-suite/. [Online; accessed 2-Jan-2020].

Cellier, F. E., and E. Kofman. 2006. *Continuous system simulation*. Springer Science & Business Media.

Chang, S. L., S. P. Cavnar, S. Takayama, G. D. Luker, and J. J. Linderman. 2015. "Cell, isoform, and environment factors shape gradients and modulate chemotaxis". *PloS one* vol. 10 (4), pp. e0123450.

Chopard, B., J. Borgdorff, and A. G. Hoekstra. 2014. "A framework for multi-scale modelling". *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* vol. 372 (2021), pp. 20130378.

Hoekstra, A. G., J. Kroc, and P. M. Sloot. 2010. *Simulating complex systems by cellular automata*. Springer.

Hu, X., A. Muzy, and L. Ntaimo. 2005. "A hybrid agent-cellular space modeling approach for fire spread and suppression simulation". In *Proceedings of the Winter Simulation Conference, 2005.*, pp. 8–pp. IEEE.

Kim, S., H. S. Sarjoughian, and V. Elamvazhuthi. 2009. "DEVS-suite: a simulator supporting visual experimentation design and behavior monitoring". In *Proceedings of the 2009 Spring Simulation Multiconference*, pp. 161. Society for Computer Simulation International.

Mayer, G. R., and H. S. Sarjoughian. 2009. "Composable cellular automata". *Simulation* vol. 85 (11-12), pp. 735–749.

Mayer, G. R., and H. S. Sarjoughian. 2016. "Building a hybrid DEVS and GRASS model using a composable cellular automaton". *International Journal of Modeling, Simulation, and Scientific Computing* vol. 7 (01), pp. 1541005.

Recktenwald, G. W. 2004. "Finite-difference approximations to the heat equation". *Mechanical Engineering* vol. 10, pp. 1–27.

Sarjoughian, H. S. 2006. "Model composability". In *Proceedings of the 2006 Winter Simulation Conference*, pp. 149–158. IEEE.

Toffoli, T., and N. H. Margolus. 1990. "Invertible cellular automata: a review". *Physica D: Nonlinear Phenomena* vol. 45 (1-3), pp. 229–253.

Von Neumann, J. 1966. *Theory of self-reproducing automata*. Urbana, University of Illinois Press.

Wainer, G. A. 2017. *Discrete-event modeling and simulation: a practitioner's approach*. CRC press.

Zeigler, B. P., T. G. Kim, and H. Praehofer. 2000. *Theory of modeling and simulation*. Academic press.

Zhang, C., and H. S. Sarjoughian. 2017. "Cellular Automata DEVS: A Modeling, Simulation, and Visualization Environment". In *Proceedings of the 10th EAI International Conference on Simulation Tools and Techniques*, pp. 11–19. ACM.

AUTHOR BIOGRAPHIES

CHAO ZHANG is a Ph.D. student in the School of Computing, Informatics, and Decision Systems Engineering (CIDSE) at Arizona State University, Tempe, Arizona. He can be contacted at zhang72@asu.edu.

HESSAM S. SARJOUGHIAN is an Associate Professor of Computer Science and Computer Engineering in the School of Computing, Informatics, and Decision Systems Engineering (CIDSE) at Arizona State University, Tempe, Arizona. He can be contacted at sarjoughian@asu.edu.

MOON G. SEOK is a Postdoctoral Fellow in the School of Computer Science and Engineering at the Nanyang Technical University, Singapore. He can be contacted at moongi.seok@ntu.edu.sg.