Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

COMPOSITION OF GEOGRAPHIC-BASED COMPONENT SIMULATION MODELS

William A. Boyd
Hessam S. Sarjoughian

Arizona Center for Integrative Modeling & Simulation
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, AZ 85281, USA

ABSTRACT

Separate simulation models (e.g., agent-based models) may depend on spatial data associated with geographic locations. Use of autonomous interaction models allows independent models to be composed into an aggregate model without alteration of the composed models. The Geographic Knowledge Interchange Broker (GeoKIB) is proposed as a mediator of spatial-temporal models. The GeoKIB regulates unidirectional interactions between composed models of the same type or not. Different input and output data types are supported depending on whether data transmission is passive or active. Synchronization of time-tagged input and output values is made possible via connections to shared simulation clocks. A spatial conversion algorithm transforms any two-dimensional geographic data map for another region of different map cell sizes and boundaries. A composition of a cellular automaton and an agent-based model is developed to demonstrate the proposed approach for spatially-based heterogeneous model composition with the GeoKIB.

1 INTRODUCTION

To create models of complex systems, it is often best to compartmentalize different aspects of systems as separate, autonomous models. These component models are each best specified with a separate set of specifications, operating with different rules and scales. Each model could be simulated independently using a simulation engine.

A model, on which simulations are based, is defined to have a particular input and output specification. The frequencies, measurement units, and meanings of the input and output variables play a significant role in a model. Some models have other specifications for I/O, such as spatial resolution or null value limitations. These unique specifications result in difficulty connecting different models, even for models that represent related systems. Differences between variable types and values must be resolved whenever the expected input of a simulation is not provided, with the same format, by the output of another simulation. While changing the simulations' I/O requirements may work in these situations, such changes are undesirable. Each simulation should be created and tested according to its own requirements, and changing an input or output violates this. Forcing adherence to a common protocol places restrictions on component models that are inappropriate to their different scopes of operation.

Multi-resolution modeling is basic to understand the dynamics of systems (Davis & Bigelow, 2008). Toward achieving this need, the composition of models has a basic role. A key challenge is to model interactions between heterogeneous model types. Interaction components can be modeled in different ways, among which is multi-formalism modeling supported with Knowledge Interchange Broker models responsible for making conversions of data types, quantity, resolution, and frequency between disparate models (Sarjoughian, 2006). Each KIB model has a syntax and semantics consistent with the disparate model types. In this paper, a variant of KIB supporting geographic maps is proposed for agent-based and

cellular automata simulations. A geographic KIB manages interactions between models with spatial and temporal differences.

The GeoKIB applies spatial data transformation for cellular automata (CA) models or spatially bounded Agent-Based Models. Cellular automata represent changes to a geographic area over time (Wolfram, 2002). For each time step, every map cell needs to read its inputs and produce its outputs as its state is determined. The updating format often varies greatly between different models. For example, considering human-landscape modeling (Barton, et al., 2016), the interactions between a model of human activities and a model of landscape dynamics may include primitive data types, such as integer and floating-point values, but also includes entire maps of data, with values spread over cells of two-dimensional maps. Particular challenges arise when transferring cellular automata data between models, especially when the models use area divisions of different boundaries and cell sizes. The need to mitigate this complexity motivated the design and development of the GeoKIB that converts two-dimensional maps to different region specifications during simulations. To aid the development and testing of the GeoKIB model, an exemplar cellular automata (CA) model for fruit growth and an agent-based model (ABM) for gatherers are composed using a geographic KIB. The CA, ABM, and GeoKIB models are created, simulated, and validated.

2 RELATED WORK

Prior work has used different approaches for the modelling and composition of cellular automaton models. These works generally are based on the well-known concept of cellular automata suitable for modeling spatiotemporal systems. A cellular automaton model is a representation of a system based on a tessellation of map cells over multi-dimension cell spaces. All cells of the automaton have the same dimensions and the same specifications for input and output space, state space, and state transitions. State transitions for a single cell of a cellular automaton are dependent on the cell's inputs, the cell's state, and the state of the cell's neighbors (Wolfram, 2002). The states of all the cells are updated with the same discrete-time behavior specification. A variety of simulators have been developed for cellular automata models. One of these is called MASON (Multi-Agent Simulator of Neighborhoods) (Luke, Cioffi-Revilla, Panait, Sullivan, & Balan, 2005). This tool is based on the agent-based modeling approach for creating, executing, and visualizing cellular automata models. Mathematica also supports run-time animation of cellular automata states, where cell behavior is determined by the mathematical formulas set for the model.

An approach called Composable Cellular Automata (CCA) for combining separate cellular automaton models has been proposed (Mayer & Sarjoughian, 2009). The approach described in this paper is closely related to the idea of CCA approach. However, it differs in that the GeoKIB emphasizes the process of model composition itself, and that the differences between the geographic representations, especially differences in spatial resolution, are handled within interaction models.

2.1 Event-Based Cellular Automata

Traditionally, the state of a cellular automaton is updated at discrete time steps. Some representations of cellular automata emphasize temporal changes. (Peuquet & Duan, 1995) describe a method of storing spatiotemporal GIS data primarily according to the time of change (event), rather than storing the full geographical state at every time interval. This organization of geographic data gives computational and conceptual advantages for geographic analyses of events, causes, and effects.

A Cell-DEVS model specification to represent each of the cells of a cellular automaton has been developed (Wainer, 2017). As DEVS-based models, component and CA models can be composed using I/O couplings. This specification models the interactions of map cells while providing a mechanism to reduce the required computation for passive cells. The performance of cellular automaton simulations was improved in this manner.

Another related research is a multiscale framework that supports modeling, simulating, and visualizing independent multiple CCAs. The framework uses I/O couplings to model interactions among CCAs. The approach in this paper, in contrast, does not support run-time component-based animation, run-time step-

by-step debugging through independent linear and superdense input, output, and state trajectories, and customized playback (Zhang, Sarjoughian, & Seok, 2020)

2.2 Model Composition

Previous work proposed a hybrid modeling approach for composing separate agent and landscape models using the Knowledge Interchange Broker (KIB). The approach proposed in this paper supports interaction modeling for composing geographic simulations. Interactions modeling is supported with mapping functions for models that can have independent timing and spatial resolutions.

A formulation for composition of cellular automata for a single group (set of cells) in which the transition function for the composition of two cellular automata is equivalent to the functional composition of the transition functions for the component automata has been proposed (Inokuchi, Ito, Fujio, & Mizoguchi, 2014). This type of composition may be useful when composing cellular automata that represent layered effects on a set of data. However, this definition of composition is not applicable to models that update independently on different data sets.

Another approach to multiscale modeling called Complex Automata (CxA) has been proposed (Hoekstra, Falcone, Caiazzo, & Chopard, 2008). A Complex Automata can be created as a composition of a set cellular automata of different spatial and temporal resolutions. Each cellular automaton of the set has a time step (Δt) and cell size (Δx) appropriate for its representation, along with temporal and spatial boundaries within the boundaries of the CxA. The CxA acts as a whole Cellular Automaton that integrates CAs that have different spatial and temporal scales. This is similar in purpose to the approach described in this paper, allowing models with different specifications and resolutions to be composed. This paper differs in that the responsibilities of managing interactions are assigned to separate, dedicated interchange components. This difference in approach was chosen to allow interaction of models that were designed separately from the other models of the composition. While a well-designed CxA can allow more efficient composition of cellular automata components designed for aggregation, a technique was required that allows future models, with different temporal formalisms (such as discrete event), to be added to an existing system. Another difference is that in this paper, two composed Cellular Automata models can have the same spatial dimensions. The CxA approach neither conceptualizes nor formalizes the Composable Cellular Automata model, which provides modularity at the level of individual Cellular Automata models for composition with component-based models (Mayer & Sarjoughian, 2009).

A methodology is proposed and implemented for a multi-resolution cellular automaton model. Its specification views cells from related models as neighbors in a hierarchical adjacency set, regardless of the differing resolutions and cell sizes (Kiester & Sahr, 2008). The implementation of the methodology requires that state transition rules for each cell be dependent on the percentage distribution of the states of the neighboring cells, rather than the number of neighboring cells that have the state. For example, in the Game of Life model (Gardner, 1970), the transitions are redefined as dependent on the ratio of neighboring cells that are alive (Kiester & Sahr, 2008). The ratio-based specification allows for interactions among models, with automatic adjustments for differing numbers of neighboring cells from other resolutions. The approach to interaction shows the assumption that all models use the same category of information greatly restricting the use of such an approach with heterogeneous models. In addition, working with a hierarchical adjacency set in this manner would most often require changing the component models and their transition rules to fit this ratio-based perspective of neighboring states.

3 INTERACTION MODELING DESIGN

The purpose of the geographic knowledge interchange broker (GeoKIB) is the composition of models with cellular spatial models. The GeoKIB is designed to allow component models to be combined into a single model when the component models have different specifications for input, output, and data structures. This design was created to be adaptable to both frequency- and event-based update types.

3.1 A Generalized Knowledge Interchange Broker Design

A composition of two or more simulations is likely to involve multiple complex interactions. A snippet of the UML class diagram of the GoeKIB design is shown in Figure 1. To manage this complexity, each of the interactions is managed by a separate component, called a KIBFunction. Each of the simulations may connect to more than one KIB to regulate its interactions with each of the other components. Each KIBFunction has the task of regulating one set of interactions between two simulations, where one simulation is designated as the input to the KIBFunction, and the other simulation is the output. When two different simulations must communicate in multiple ways, multiple KIBFunction instances can be used as brokers between those same two simulations. In particular, when two models must send information to each other, at least one KIBFunction instance is needed to regulate the data flow in each direction.

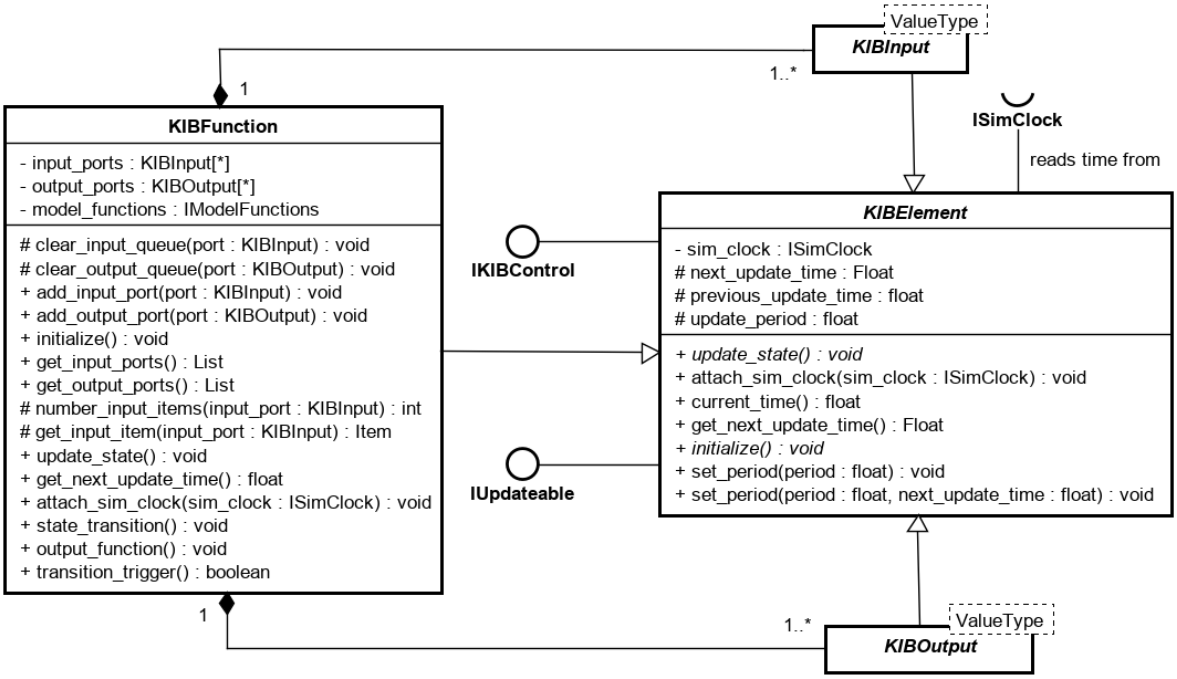

Figure 1: A Partial Class Diagram Design for GeoKIB

A Each KIBFunction acts as a separate model. The inputs, outputs, transition functions, and timing of each KIBFunction must be defined to produce the correct outputs at the correct times in response to the received inputs. This is done by defining the abstract methods state_transition() and is_transition_triggered(). The state_transition() method acts as the main state transition function of the KIBFunction object, including the acts of dequeuing inputs, transforming values, and enqueuing outputs. The method is_transition_triggered() returns a Boolean value that demines whether the state_transition() method should be called. As an example, is_transition_triggered() may be defined to return true only when a new input is available, indicating that state_transition() will not be activated when no input has been received.

For a KIBFunction object to have the ability to synchronize its timing with other components or the composed simulation, it needs the ability to read the time from an external source. For this purpose, a KIBFunction can read the current simulation time from an external clock via the ISimClock interface. KIBFunction uses the method get_current_time() method from the ISimClock interface to obtain the current

simulation time. Different simulations and KIBFunction objects may use the same or different clocks for their timings.

A KIBFunction object receives inputs from dedicated input ports and sends outputs through assigned output ports. Each of the ports is parameterized to specify the data type of the values that are sent through the port. Each of the ports uses a queue to store and retrieve values. Each received value at an input port is tagged with the time of receipt. In an output port, values are tagged with a scheduled time for the output.

Different types of ports can be created and attached to a KIBFunction. A discrete-event input port can receive inputs at any time. Upon receiving an input value, the port alerts the KIBFunction of the external event by calling the state_transition function. A discrete-time input port can be set to read an input value at specified time intervals. A passive output port allows the KIBFunction's output values to be read at any time, allowing other models to check values when required. An active output port delivers outputs to a specified model, allowing event inputs to be delivered to a model. This variety of ports allows one KIBFunction unit to connect to both a discrete-event model and a discrete-time model when necessary.

The KIBFunction method update_state() is used as a "wakeup" function. When the method is called, the object checks whether any updates are needed, performs any scheduled actions, then updates its state accordingly. Calling update_state() more than once does not cause any inconsistencies in the KIB's functionality or state, so the method may be called at any time to ensure that the state is current. In the class KIBElement, update_state() is an abstract method, meaning that it has no "default" implementation and uses polymorphism to provide the implementation appropriate for the subclass.

The implementation of update_state() depends on the class and its specific purposes. In KIBFunction, update_state() first updates each of the attached input ports. Any scheduled inputs are taken by the input ports. Then, if the is_transition_triggered() method returns true, state_transition() is called. Finally, values at all output ports are updated, and values are sent to other models if scheduled.

3.2 Geographic Interaction Model Design

The design of the KIBFunction allows different data types or classes for inputs and outputs. The input and output ports can be parameterized to operate together with any set of models. While this is compatible with geographic models, it is best to extend the design of KIBFunction to address the needs specific to geographic maps.

The basic data structure for values spread across a geographic area is a geographic map, represented as an array of values associated with map locations. For this project, the GeoMap class was created for this representation. Each GeoMap object has one region specification, which defines the boundary values and divisions, or cells, of the region. Each cell of the GeoMap holds one value, which may be null. The values are stored as a two-dimensional array in memory.

The class GeoKibFunction inherits from the KIBFunction class and adds functionality specific to working with geographic transformations between models. A GeoKibFunction is assigned a separate region for its inputs and outputs. The differences between the MapRegion objects are used as the basis for the spatial conversions that are performed within the class. As with its parent class, GeoKibFunction is an abstract class, meaning that it cannot be directly instantiated. The abstract methods from KIBFunction, state_transition() and is_transition_triggered(), are not defined in GeoKibFunction and must be defined in a specialized extension of the class.

GeoKibFunction provides the spatial_conversion() method, which converts a GeoMap from the input region to the output region. The spatial_conversion() method accepts one GeoMap object as a parameter, which must match the source region of the GeoKibFunction. Within spatial_conversion(), a two-dimensional array of values is created with the number of row and columns specified in the destination_region attribute of GeoKibFunction. Each element of the new array corresponds to a geographical area, with values that can be determined for its north, south, west, and east boundaries of a destination map cell. These boundaries can then be matched with the original GeoMap to find an overlapping area. Because of differences in resolution, the destination cell may not align with the

boundaries of cells from the source map. It is possible for a destination cell to overlap multiple cells from the source map in different area proportions, as shown in Figure 2.

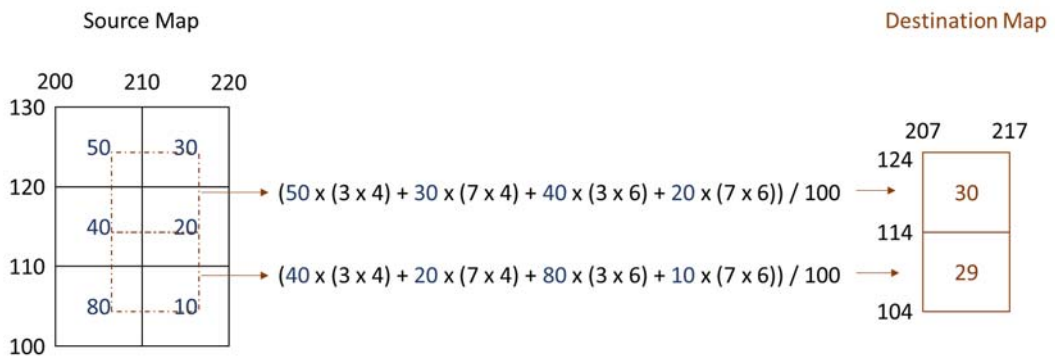

Figure 2: Example of the conversion to a region with differently aligned cell boundaries.

The value of one destination cell in spatial_conversion() is determined by finding an average of the values from the corresponding (overlapping) area of the source map, weighted based on area sizes. For each of the source map cells, its value is multiplied by the size of the area of overlap between the source cell and destination cell. The products are added from all the corresponding source cells to create a weighted sum. Similarly, the sizes of the overlapping areas are added to find a total area. The weighted sum is divided by the total area to find the weighted average value for the destination cell.

3.3 Model Integration

Creating a composition of models involves defining the characteristics of the composition, setting up the component models and interaction models, configuring the connections, and synchronizing the execution times. To verify that all of these steps could be applied, two simple exemplar models were created with geographic components. A fruit growth model was developed in which the number of fruit that reproduce in an area is proportional to the number of nearby fruit. A separate gatherer model has gatherer agents explore or settle in a landscape to obtain food.

Composition of the fruit growth model and the gatherer model requires that the interaction specifications be defined. The gatherer model requires an input of the amount of food available at each location of the map. This input is specified to be the same as the number of fruit at the same location, taken from the fruit growth model. The input to the fruit growth model is the change, at each location, in the number of fruit due to external influences. This is determined by multiplying the number of gatherers at the location by the rate of consumption by -1. In the composition, it is assumed that different regions and time steps are used by the different models, so such differences must also be managed.

For the components that deal with interactions between the fruit growth model and gatherer model, extensions of the GeoKibFunction class were made. One instance of GeoKibFunction was made to take a map of the number of fruit as input and produce a map of the amount of available food as output. Although these amounts are equal, differences in region and time scaling needed to be addressed. The fruit growth model outputs the current number of fruit, so there is no case in which multiple inputs should be combined by addition or any other operations. Therefore, if multiple input maps are received before an output map needs to be sent, the output map is processed from only the most recently received input. If multiple outputs are needed before new inputs are received, the resulting map is output multiple times.

During a simulation of the composed models, the simulation manager must determine when each of the component simulations must be progressed. Each component simulation can be queried for the time step for which it was last run. Information about the time of the last step and length of the time steps can be used

to determine the time to run the next step for each component. The next time step for the composition is the least of the next times for the components. At each time step, the simulation manager finds the components whose next scheduled time step is equal to the current time, and commands those components to progress for one time step. It is important to note that this synchronization protocol is only possible for components that can be externally controlled to progress one step at a time.

4 DEMONSTRATION OF INTERACTION MODEL EXECUTION

A demonstration is shown here of the interactions between the exemplar models via the GeoKIB. For this scenario, the growth model and the gatherer model are run in a joint simulation. The regions for the two models have same outer boundaries, but different resolutions. The simulation takes place in a 30km x 20km area, with west and east boundaries labeled 300 and 330, and south and north boundaries labeled 200 and 220. The simulation area for the fruit growth model is divided into 24 square cells that each have size 5 x 5. The same area in the gatherer model is divided into 6 square cells of size 10 x 10. Each gatherer eats, when available, four fruit per time step. A gatherer can see cells up to two spaces away and can move two spaces within a time step.

The state of the combined simulation is updated at integral time steps. In this simulation, the state of the gatherers is updated every time cycle, but the state of the fruit growth is only updated for even numbered time steps. Time 0 is the start time of the simulation, and changes to the state occur after, not during, time 0. To avoid cyclic data dependencies, data from a component model can only be used by the other component model at a time later than the time of data production. That is, the number of fruit at time 2 influences the gatherer model at times 3 and 4, but not at time 2. This requirement allows the results of the simulation to be independent of the order of execution of the component models.

At the beginning of the simulation (time 0), the KIB takes the map information about the number of fruit from the fruit growth model. The KIB uses the data to create, as outputs, maps of available food for the gatherer model for times 1 and 2. The KIB must convert the spatial resolution to a format appropriate for the gatherer model. The food is cumulative data, so the values in the cells of the output maps are obtained from sums, not averages, from the corresponding input areas, as shown in Figure 3. Since the fruit growth model is not updated before time 2, identical maps are produced by the KIB for times 1 and 2.

As the fruit growth simulation has a period of 2 time units, its first iteration of execution is at time 2. The KIB is responsible for providing data to the fruit growth model concerning the changes in the number of fruit due to external influences, including the amount eaten by the gatherers.

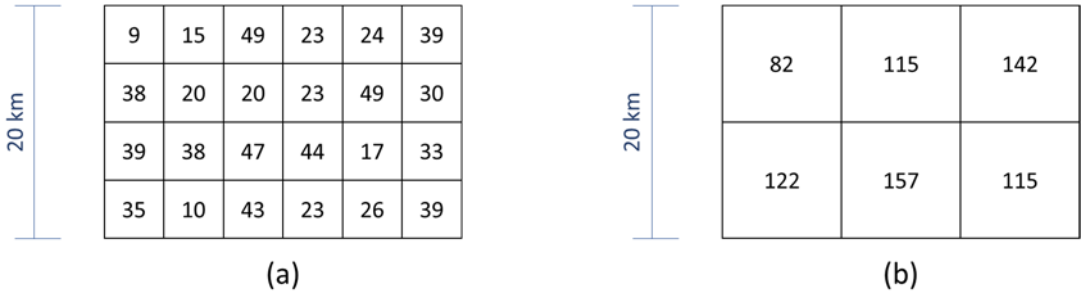

Figure 3: Conversion of spatial resolution by the KIB. (a) Fruit growth model initial numbers of fruit. (b) KIB output food maps for times 1 and 2.

Also at time 0, the KIB receives a map that tells the locations of the gatherers. The KIB produces maps that tell external changes to the number of fruit at each map cell. Since the KIB receives fruit values more frequently than it sends food values, the KIB needs to collect multiple inputs maps and assemble them before it sends each food map to the gatherer model.

At time 1, the first time step of the gatherer simulation is executed. During this step, the two gatherers examine their surroundings to find the cell with the largest amount of food. The cell with 157 food has more food than any other cell, and it is visible to both gatherers. The cell becomes the target of both gatherers. They each decide to move to the cell during the following time step. Because they do not move during time 1, the output from the gatherer simulation (shown in Figure 4) is the same as the output at time 0.The KIB takes this map of numbers of gatherers during time step 1. The KIB has the inputs it needs to create its output, a map of the changes in fruit amounts, for use by the fruit growth simulation at time 2. The KIB must complete a few steps to create each of its outputs for the fruit growth simulation.

1. Collect and store each output from the gatherer simulation. Each output is a map of the numbers of gatherers, using the spatial resolution of the gatherer simulation.
2. Combine input maps received between outputs by adding the values at corresponding locations over each input map. The summation gives the total number of gatherer visits per location, in the gatherer simulation's spatial resolution.

Figure 4: Input maps of numbers of gatherers processed by the KIB at the start of time 2. (a) Numbers of gatherers at time 0. (b) Numbers of gatherers at time 1.

3. Create a map of fruit change by multiplying the gatherer visits by the gatherers' appetite (fruit eaten per time step), multiplied by -1. The resulting map indicates the changes in the amounts of fruit between updates in the fruit growth simulation (shown in Figure 5). The values are all negative to indicate that the fruit was removed by the gatherers.

Figure 5: Creation of the fruit change map by the KIB at the start of time 2. (a) Sum of input gatherer maps for times 0 and 1. (b) Change in fruit at time 2, using the spatial resolution of the gatherer simulation.

6. Perform a spatial resolution conversion on the fruit change map to convert its region from that of the gatherer simulation to that of the fruit growth simulation (see Figure 6).

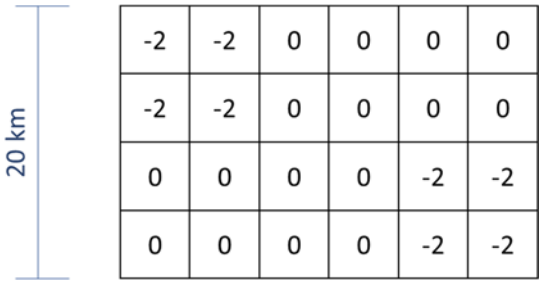

Figure 6: Change in fruit at time 2, converted to the spatial resolution of the fruit growth simulation.

7. Make this output map, of the change in fruit, available for the fruit growth simulation at time 2.

At time 2, the fruit growth simulation can proceed to execute its first simulation step, once it receives as input the map of changes in fruit from external influences. The simulation can continue to run with this procedure, allowing both the fruit growth simulation and the gatherer simulation to progress without concerns of the other simulation's differences. With similar procedures, the KIB can run combined simulations when the fruit growth simulation runs more frequently than the gatherer simulation, or when the spatial resolution of the gatherer simulation is higher than that of the fruit growth simulation.

5 TESTING AND RESULTS

The principles of the Geographic Knowledge Interchange Broker (GeoKIB) were applied to connect the two exemplar models: the fruit growth and gatherer models. In order to verify the GeoKIB's ability to compose models at different resolutions, a set of different simulations using the models was run. The different simulations differ in the relative spatial and temporal resolutions of the two exemplar models. Distances within simulations represent meters. All simulations of the set use the same set of spatial boundaries, with south boundary at a value of 1000, north boundary at 1320, west boundary at 5000, and east boundary at 5640. Each simulation time step represents one day. Within the simulations of the fruit growth model, the number of fruit at each cell increases by an amount determined by multiplying the number of fruit in the surrounding 5 meters by 0.010. A maximum fruit density of 0.3 fruit per square meter is allowed. Gatherers can live for a maximum of 30 days, and the 1000 initial gatherers have an age of 10 days. A gatherer can reproduce and create another gatherer after staying settled for 5 days. Each of the gatherers needs to consume 4 fruit per day. A gatherer will die if there is not enough available food for 3 days.

Initial values were provided to the exemplar models as maps that indicate the locations of fruit and gatherers over the simulated landscape. To remove stochasticity from the simulations, the same map of fruit locations and the same map of gatherer locations were used to create the initial state for each of the simulations. In cases where a model had a different spatial resolution than the map of initial locations, the locations were converted to create a state most consistent with the given initial locations using the model's resolution.

The first of these simulations has both models use the same spatial and temporal resolutions. Both models use a map resolution of 32 rows and 64 columns, for a total of 2,048 cells. The models each update once per simulation time step during the 100 total time steps. Other simulations had varying differences in spatial resolution between the exemplar models, while keeping the temporal resolutions equal. In these simulations, one model had the higher resolution with 32 rows and 64 columns, while the other model represented the same area with fewer cells. Additional simulations retained the same spatial resolutions for the two models while using different time steps between updates for each of the models. In these, the state of one of the models was updated during each simulated day, while the other was updated only after a given number of days had passed. Finally, simulations were run in which spatial and temporal differences existed

between the two exemplar models. These simulations included all combinations with spatial area resolution factors of 4 or 16 and temporal resolution factors of 2, as shown in Table 3.

Because the GeoKIB was designed to be flexible regarding the differences in spatial and temporal resolutions, it was able to connect the different simulations for all of the configurations. The GeoKIB automatically determined how to manage the differences in data input and output timings when given inputs that indicate the sizes of the time steps for each of the simulations.

Table 3: Configurations for simulations with differences in both spatial and temporal resolutions.

Fruit Number of Cells/Gatherer Number of Cells Ratio	4:1	4:1	16:1	16:1	1:4	1:4	1:16	1:16
Fruit Number of Time Steps/Gatherer Number of Time Steps Ratio	2:1	1:2	2:1	1:2	2:1	1:2	2:1	1:2
Fruit Number of Rows	32	32	32	32	16	16	8	8
Fruit Number of Columns	64	64	64	64	32	32	16	16
Gatherer Number of Rows	16	16	8	8	32	32	32	32
Gatherer Number of Columns	32	32	16	16	64	64	64	64
Fruit Period	1	2	1	2	1	2	1	2
Gatherer Period	2	1	2	1	2	1	2	1

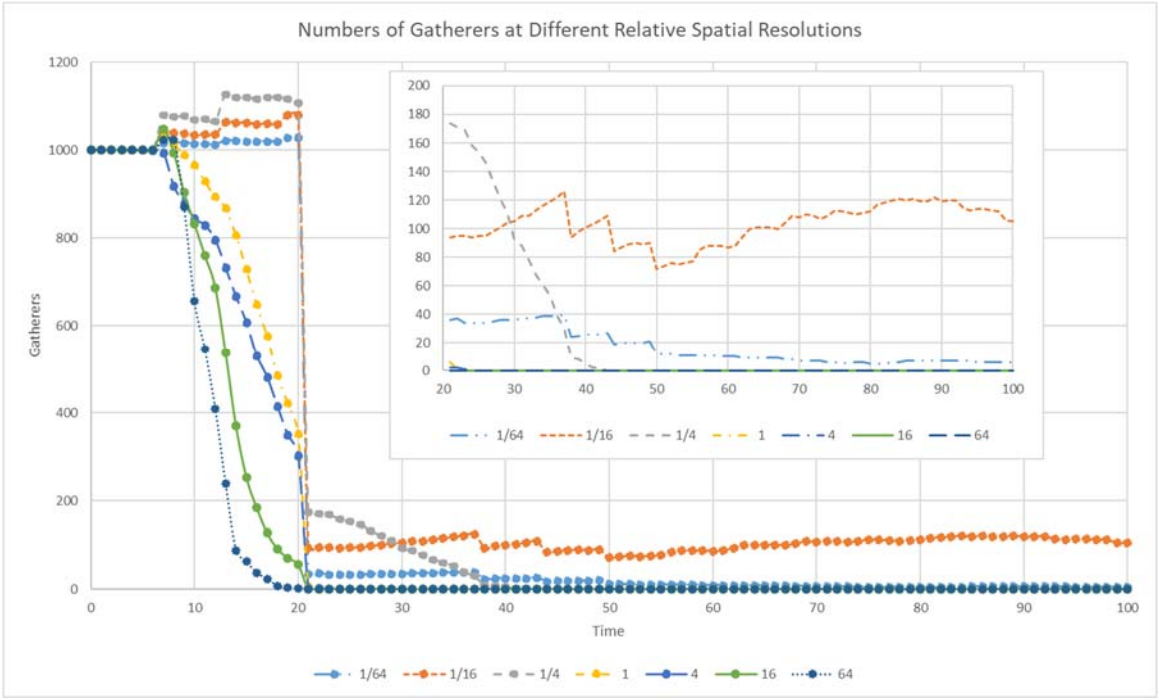

Figure 34: Number of gatherers during simulations with different spatial resolutions for the gatherer and fruit growth models. Both the gatherer and fruit growth models have temporal resolution of 1 update per day. The key shows the number of gatherer cells divided by the number of fruit growth cells.

Results of interactions between gatherers and fruit are apparent at all resolutions. The large numbers of gatherers, along with the limited number of fruit, result in a limit in the growth of the populations. The

number of fruit decreases over time wherever gatherers are present, and gatherers tend to collect in areas with large numbers of fruit. As all of the interactions are managed by the GeoKIB, this shows the success of the GeoKIB in exchanging information between the models. Figure 34 shows the effects of changing spatial resolutions on the number of gatherers.

The differences in relative spatial resolutions resulted in some differences in results. In particular, the behavior of the gatherers was especially sensitive to spatial resolution. At lower relative resolutions of the gatherer model, each map cell of the gatherer model corresponds to multiple cells of the fruit growth model. This results in more fruit available at each of the gatherers' cells, making each of the cells more sustainable. However, the lower number of cells for the gatherers also results in fewer locations to settle and faster depletion of food.

A limitation of this approach to multi-resolution modeling became apparent. When models rely on other models for data, differences in resolution can result in missing or insufficient information for one of the models. As an example, when the gatherer model was updated 5 times as often as the fruit growth model, the GeoKIB needed to supply information to the gatherer model from as much as 5 days previous to the current day. Then, after the fruit growth model was updated, it was impacted by all 5 days of influences from the gatherers at once.

6 CONCLUSION

In this paper, a proposed hybrid modeling approach for combined spatial-temporal interaction model design is developed. A unique advantage of this hybrid model composition approach is the capability to define interactions between models of arbitrary resolution levels. This is useful since interactions amongst natural, social, and built systems are complex, especially for systems that have multiple geographic resolutions. Interaction models can be used to understand or design complex structures and behaviors for systems that exhibit different temporal and spatial scales and configurations. When models communicate their geographic data via GeoKIB, the data can be transformed to account for differences in boundaries and resolution, as well as the differences in measurements and timing protocols. Interaction modeling with geographic mapping algorithms can be used to transform values. Interaction models allow exact or approximate calculations for different geographic regions in a transparent and disciplined manner. For future work, visual support for model development of GeoKIB is anticipated. Combining the rigor of model specification with complementary capabilities such as visual modeling and high-performance computing can offer a new level of access and understanding into the nexus of biological and cyber-physical systems, among others.

ACKNOWLEDGMENTS

This research is supported by the National Science Foundation (NSF grant #DEB-1313727).

REFERENCES

Barton, C. M., Ullah, I. I., Bergin, S. M., Sarjoughian, H. S., Mayer, G. R., Bernabeu-Auban, J. E., & Arrowsmith, J. R. (2016). Experimental socioecology: Integrative science for anthropocene landscape dynamics. *Anthropocene*, 34-45.

Davis, P. K., & Bigelow, J. H. (2008). *Experiments in Multiresolution Modeling (MRM).* Santa Monica, CA, USA: RAND.

Gardner, M. (1970). Mathematical Games - The fantastic combinations of John Conway's new solitare game "life". *Scientific American, 223*(4), 120-123.

Hoekstra, A. G., Falcone, J.-L., Caiazzo, A., & Chopard, B. (2008). Multi-scale Modeling with Cellular Automata: The Complex Automata Approach. *8th International Conference on Cellular Automata for Research and Industry. 5191.* Yokohama, Japan: Springer.

Inokuchi, S., Ito, T., Fujio, M., & Mizoguchi, Y. (2014). A formulation of composition for cellular automata on groups. *IEICE TRANSACTIONS on Information and Systems, 97*(3), 448-454.

Kiester, A. R., & Sahr, K. (2008). Planar and spherical hierarchical, multi-resolution cellular automata. *Computers, Environment and Urban Systems, 32*(3), 204-213.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., & Balan, G. (2005). Mason: A multiagent simulation environment. *Simulation, 81*(7), 517-527.

Mayer, G., & Sarjoughian, H. (2009). Composable cellular automata. *Simulation, 85*(11-12), 735–749.

Peuquet, D. J., & Duan, N. (1995). An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data. *International journal of geographical information systems, 9*(1), 7-24.

Sarjoughian, H. S. (2006). Model Composability. *Proceedings of the 38th Conference on Winter Simulation*, (pp. 149-158).

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

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

Zhang, C., Sarjoughian, H. S., & Seok, M. G. (2020). A Framework for Composable Cellular Automata DEVS Modeling, Simulation, and Visualization. *Theory and Foundations of Modeling and Simulation, SpringSim.* Baltimore.

AUTHOR BIOGRAPHIES

WILLIAM A. BOYD earned his Master's Degree in Computer Science in the School of Computing, Informatics and Decision Systems Engineering (CIDSE) at ASU, Tempe, AZ, USA. He can be contacted at waboyd@asu.edu.

HESSAM S. SARJOUGHIAN is an Associate Professor of Computer Science & Computer Engineering at Arizona State University and Co-Director of the Arizona Center for Integrative Modeling and Simulation. His research focuses on model theory, polymorphic model composability, distributed co-design modeling, visual simulation modeling, real-time modeling, and service-oriented simulation. He can be contacted at sarjoughian@asu.edu. His website is http://sarjoughian.faculty.asu.edu/.