A Composable Discrete-Time Cellular Automaton Formalism

Gary R. Mayer[†] and Hessam S. Sarjoughian[*]

[†]Gary.Mayer@asu.edu and [*]Sarjoughian@asu.edu
Arizona Center for Integrative Modeling and Simulation, School of Computing and Informatics, Arizona State University, Tempe, AZ

Abstract Existing Cellular Automata formalisms do not consider heterogenous composition of models. Simulations that are grounded in a suitable modeling formalism offer unique benefits as compared with those that are developed using an ad-hoc combination of modeling concepts and implementation techniques. The emerging and extensive use of CA in simulating complex heterogeneous network systems heightens the importance of formal model specification. An extended discrete-time CA modeling formalism is developed in the context of hybrid modeling with support for external interactions.

1 Introduction

Cellular Automaton modeling has been applied in many ways across many interesting problem domains. A majority of these use the automaton to represent the entire modeled system [1,4]. However, to simulate complex heterogeneous systems, other kinds of models such as rules-based agents are necessary. An example system is one that can model disaster relief efforts. A simulation model for such a system may consist of people, response teams, landscape, and facilities. These can be described using rules-based agent and CA modeling approaches. Synthesis of these different kinds of sub-system models results in a *hybrid* modeled system. A hybrid model is a model of a system, comprised of two or more distinct sub-systems (*e.g.*, [9]). The term "hybrid" is used to imply that each of the sub-system models can be developed with its own appropriate structural and behavioral details. The significance of hybrid modeling is to overcome the unsuitability or impracticality of a monolithic model for describing the disparate types of dynamics (*e.g.*, describing the interactions between fire crews and the spread of fire).

To integrate different kinds of models, a taxonomy of modeling approaches has been proposed. One of these is poly-formalism [6, 10]. In the poly-formalism approach, a separate model is proposed to describe the interactions between the sub-system models. Unlike all other existing model composability approaches, poly-formalism requires a formulation of the models' interactions. This approach specifies the composition of two disparate modeling formalisms via a third modeling formalism. What is gained is a a concise delineation between the sub-system mod-

els and their interaction via an *interaction model*[1], which is also called Knowledge Interchange Broker. However, in order to use a CA as a sub-system model within the poly-formalism composed hybrid system model, a CA formalism must distinguish between the interaction between the cells within the CA and between the system as a whole and the interaction model. A CA formalism needs to support the specification of its external input/output with another modeling formalism vice as a black box model. This capability is lacking from current CA specifications.

Model composition aside, it is useful for formal CA models to be implemented in tools such as Geographic Resources Analysis Support System (GRASS) [3]. This requires developing a mapping from CA models to the GRASS implementation constructs. Conversely, a mapping from the GRASS implementation constructs to CA models can enable formalizing the implementation of CA.

To enable the CA to interact with other non-cellular systems in a systematic fashion, the authors have devised a general, domain-neutral description (*i.e.*, formalism) of a multi-dimensional cellular automaton, including its input and its output. The formalism is a network representation built upon the two-dimensional multi-component Discrete-Time System Specification provided in [13]. The base formalism has been expanded to represent the input from and output to other models[2]. It is devised as a 3-dimensional network which may easily be paired down to either a 1-dimensional or 2-dimensional network system.

Considering the system structure from the perspective of a network, it is possible to conceive many different configuration variations — hexagonal, triangular, octahedron, and irregular networks, for example. The formalism defined here could be extended to any regular tessellation of cells, regardless of the "shape" that defines the tessellation. However, for simplicity in describing the formalism, only a grid or cube structure will be explicitly defined. Regardless of network configuration, the connectivity of the network is restricted to immediate neighbors. The cells to which a cell is connected are referred to as its *neighborhood*. A cell's state transition is influenced by its neighborhood and, possibly, by its own current state. The cells which influence the state transition are called the cell's *influencers*. The difference between the set of influencers and the set of cells representing the neighborhood is that the influencers may contain the cell itself.

2 Composable CA Formalism

A *formalism* defines its system using a domain-neutral *specification* and *execution*. The specification is a mathematical description of the system defining structure and behavior. The execution portion defines machinery which can execute model descriptions (*i.e.*, an abstract simulator). A three-dimensional, composable CA, represented by a network, N, can be described by a quintuple:

[1] Discussed in more detail in Section 3.

[2] The intent is to compose with non-cellular models but two CA may be mapped as well as described in the example in Section 2.

$$N = \left\langle X_N, Y_N, D, \{M_{ijk}\}, T \right\rangle, \text{ where} \tag{1}$$

$X_N = \{\, \overline{X}_{ijk} \,|\, (i,j,k) \,\} \vee \emptyset$ is the set of input values mapped to the network, N,

$Y_N = \{\, \overline{Y}_{ijk} \,|\, (i,j,k) \,\} \vee \emptyset$ is the set of external output for all M_{ijk} from the network structure, N,

$T = \{\, h_m \,|\, 0 \leq m \leq n \,\}$ is the time-ordered, finite set of time intervals, h_m, (*i.e.*, $\{h_0, h_1, h_2, \ldots, h_n\}$) such that $m, n \in \mathbb{N}_0, \mathbb{N}_0 \equiv (\mathbb{N} \cup \{0\}) - \{\infty\}, h_m \in \mathbb{N}^*$, $\mathbb{N}^* \equiv \mathbb{N} - \{0, \infty\}$ and $\forall h_m, h_m$ occurs before h_{m+1} (*i.e.*, h_0 occurs before h_1, h_1 occurs before h_2, etc.),

$D = \{\, (i,j,k) \,|\, a \leq i \leq b, c \leq j \leq d, e \leq k \leq f \,\}$ is the index set where $a, b, c, d, e, f \in \mathbb{Z}$; the total number of components, $|\{M_{ijk}\}|$, assuming a regular, contiguous network, is $((b-a)+1) \times ((d-c)+1) \times ((f-e)+1)$; and $\forall (i,j,k)$, the component M_{ijk} is specified as

$$M_{ijk} = \left\langle X_{ijk}, Y_{ijk}, Q_{ijk}, \mathrm{I}_{ijk}, \delta_{ijk}, \lambda_{ijk}, T \right\rangle, \text{ where} \tag{2}$$

$X_{ijk} = \dot{X}_{ijk} \cup \overline{X}_{ijk}$ is an arbitrary set of input to M_{ijk}, where $\dot{X}_{ijk}$ is the input into M_{ijk} originating from the set of output of its influencers, I_{ijk}, and $\overline{X}_{ijk} \subseteq X_N$ is the input originating from outside the network, N, which is mapped to M_{ijk}. Note that $\overline{X}_{ijk}$ may be $\emptyset$,

$Y_{ijk} = \dot{Y}_{ijk} \cup \overline{Y}_{ijk}$ is an arbitrary set of 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 $\overline{Y}_{ijk} \subseteq Y_N$ is the output from M_{ijk} that contributes to the network output, Y_N. Note that $\overline{Y}_{ijk}$ may be $\emptyset$,

Q_{ijk} is the set of states of M_{ijk},

$\mathrm{I}_{ijk} \subseteq D$ is the set of influencers of M_{ijk}; may include M_{ijk}; and $|\mathrm{I}_{ijk}| = 2$ or 3 if both j and k are constant (1-dimensional network), $|\mathrm{I}_{ijk}| = 8$ or 9 if just k is constant (2-dimensional network), or $|\mathrm{I}_{ijk}| = 26$ or 27 if neither i, j, or k are constant (3-dimensional network), the different values for each being indicative of whether or not M_{ijk} acts as its own influencer; furthermore, for any dimension network, $|D| \geq$ the larger of each value for each type of network (*i.e.*, 3, 9, and 27 for a 1-D, 2-D, and 3-D network, respectively),

$\delta_{ijk}(t_r) : \underset{\iota \in \mathrm{I}_{ijk}}{\times} \dot{X}_{ijk}(t_r) \times \overline{X}_{ijk}(t_r) \times Q_{ijk}(t_r) \rightarrow Q_{ijk}(t_{r+1})$ is the state transition function of M_{ijk} at time t_r, which is dependant upon the current state,

$Q_{ijk}(t_r)$, to map the set of influencer input, $\underset{\iota \in \mathrm{I}_{ijk}}{\times} \dot{X}_{ijk}(t_r)$, and the external input, $\overline{X}_{ijk}(t_r)$, to the new cell state at time $t_{r+1}, Q_{ijk}(t_{r+1})$,

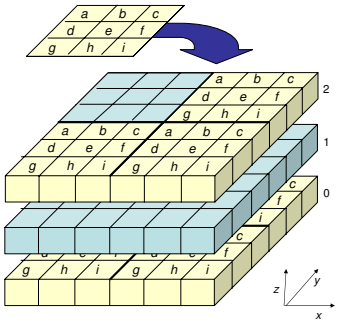
$\lambda_{ijk} : Q_{ijk} \rightarrow Y_{ijk}$ is the output function of M_{ijk} at time t_r that maps the cell state, Q_{ijk}, to the component output, Y_{ijk}, and

T is the network time-ordered set of finite time intervals (defined above) such that $\forall r, 0 \leq r \leq |T|-1, \{\, \delta_{ijk}(t_r) \Rightarrow \Delta t \equiv t_{r+1} - t_r = h_r \in T \,\}$; $r, t \in \mathbb{N}_0$; $h_r \in \mathbb{N}^*$; t_r is the time at the current state transition, $\delta_{ijk}(t_r)$; t_{r+1} is the time at the next state transition; h_r is the time interval between t_r and t_{r+1}; and the time when the network, N, is in its initial state, q_0, is represented by t_0.

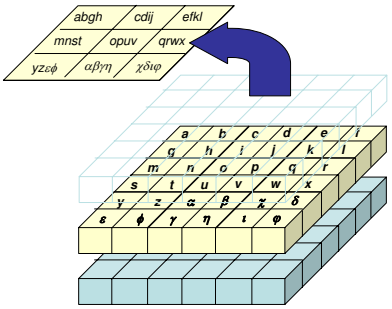
The component M_{ijk} defined in Eq. (2), which represents a cell within the CA, requires that the output function, λ_{ijk}, occur immediately before the transition function, δ_{ijk}, takes place. Using this approach, if M_{ijk} is specified to be its own influencer, then its output (dependent upon its current state) influences its next state. Otherwise, the next state of M_{ijk} is dependent solely upon the output of the surrounding cells defined to be influencers of M_{ijk}. Also, while the time interval between discrete-time segments is not necessarily constant, the above equation specifies that the difference between any two specific segments, Δt, must be the same for all components, M_{ijk}, within the network at any given time, t_r. In other words, all components within the network execute their output function and undergo state transition at the same scheduled time. This should be apparent from the fact that the network time-ordered, finite set of time intervals, T, is a part of the septuple that defines each component cell within the network.

Note that input and output from the CA is a set of values for each cell within the CA structure. This implies that there must exist a mapping between the input and the CA structure. Similarly, the output has the structure of the network but may then be mapped to whatever form the recipient requires. This mapping is unidirectional and, if between two CA, does not require that the two CA have the same structure. For example, a two-dimensional CA of size 3×3 may be mapped to a three-dimensional CA of size $6 \times 6 \times 3$. The output of the 3×3 network, $Y_{N(3\times3)}$, may be mapped to the input of the $6 \times 6 \times 3$ network, $X_{N(6\times6\times3)}$, by applying the output matrix six times as three quadrants to two of the three z layers of the larger network (see Fig. 1(a)). Note that, in this example, one quadrant of layers 0 and 2 and all of layer 1 of the $6 \times 6 \times 3$ network will not receive external input (*i.e.*, $\overline{X}_{ijk} = \emptyset$). Similarly, the output from the $6 \times 6 \times 3$ network may be mapped to the smaller network as shown in Fig. 1(b). For this example, only the output of the second layer will be mapped to the 3×3 network by partitioning the layer of the $6 \times 6 \times 3$ structure into nine 2×2 sub-networks. For the other layers, $\overline{Y}_{ijk} = \emptyset$. The corresponding output from the cell within each sub-network is then aggregated and provided as input to the respective cell in the smaller network. The input and output in this example

is made distinctly different to stress the point that the mapping may be arbitrarily revised as necessitated by the domain. This also highlights a key difference between a multi-dimensional CA and multiple CA which are mapped to one another.



(a) Mapping 2D external output to 3D external input. Cells without letters indicate no external input is received.



(b) Mapping 3D external output to 2D external input. Cells without letters indicate no external output is generated. (Top layer outlined for clarity.)

Fig. 1 An example of mapping two cellular automata

A cellular automaton, regardless of the number dimensions, is a system comprised of homogeneous components. All internal input and output must not be $\emptyset$. On the other hand, two CA mapped to one another may differentiate in many ways — structure, possible states, etc. How they relate to one another is arbitrarily defined via the mapping. Furthermore, the external output from one cellular automaton, Y_N, (and, therefore, the external input, X_N, to the other) may be $\emptyset$.

One other item to consider is edges. In the most simple of cases, the CA may be treated as a torus, in which case, there are no edges. However, while this eases the need to specifically address edges, it does not allow the CA to faithfully represent a realistic landscape. While this paper does not specifically address an execution mechanism for this formalism, it should be said that for edge conditions, it is best that no function explicitly anticipates the number of influencer input values. Instead, it should use the cardinality of the input set. The above formalism does provide the capability to explicitly manage values going into and being output from edge cells. For example, if the CA was being used to model watershed across a landscape, the external input, X_N, could map the specific values entering the edge cells while the external output from each cell, Y_N, could provide the data for material departing the cell (*i.e.*, off of the edge).

Proper execution of the formalism requires more than the state transition and output mechanisms provided in the specification. By specifying that the output function, λ_{ijk}, for each component, M_{ijk}, is solely dependent upon the current state, Q_{ijk}, to produce the output, Y_{ijk}, a delay is introduced which prevents a delayless, infinite feedback loop between components. The use of discrete-time (which does not

allow zero-time state transitions) supports the assurance that there will not be a delayless feedback loop once state transitions do occur. Next, a mapping approach for handling the input and output between the cells and the network and between the network and any external system is provided. This section then concludes with an algorithm which specifies the steps in the execution of the specification.

While the network is the system of cells, for the purposes of understanding the mapping, it may be easier to at first think of the network, N, as a shell around the set of cells, $\{M_{ijk}\}$, which comprise the system. Then, one can think of a mapping function which aggregates the individual cellular output, $\overline{Y}_{ijk}$, to the network output, Y_N. Similarly, another mapping function could provide the opposite, disaggregation of data from the network input, X_N, to the respective external input for each cell, $\overline{X}_{ijk}$. The functions $f_{out} : \bigcup_{(i,j,k)\in D} \overline{Y}_{ijk} \rightarrow Y_N$ and $f_{in} : X_N \rightarrow \bigcup_{(i,j,k)\in D} \overline{X}_{ijk}$ are the aggregation and disaggregation functions which provide external output from and input to the cells, respectively. Next, consider that an external system will likely have a different structure than the CA, may not provide external input to the entire CA structure, may not have external input every execution cycle, and may not receive output from every cell in the network. It would be inefficient and, from a component visibility perspective, inappropriate, for the external system to map to every cell in the network. Thus, two other mapping functions are required. Representing the external system with the symbol, Θ, the output from the network may be given as $g_{out} : Y_N \rightarrow \Theta_{input}$ and the input from the external system to the network as $g_{in} : \Theta_{output} \rightarrow X_N$. Function composition may then be used to define the entire mapping from the external system, Θ, to the cells and back. In other words, $g_{out} \circ f_{out} : \bigcup_{(i,j,k)\in D} \overline{Y}_{ijk} \rightarrow \Theta_{input}$ and $f_{in} \circ g_{in} : \Theta_{output} \rightarrow \bigcup_{(i,j,k)\in D} \overline{X}_{ijk}$.

While it may at first appear to have been more efficient to employ two functions which map the external system and cells directly, the reader is reminded that the purpose of this work is to present a "composable CA" which is part of a larger system. To draw away from a monolithic system and to provide flexibility in design, interaction between systems, and reuse, the concept behind the four mapping equations should be employed. These draw a distinct delineation between what occurs within the network and outside of it. Furthermore, it allows the external system to use its own unique identifiers for cells without specific knowledge of the network's internal indexing structure. Referring back to Fig. 1(a) as an example, if the 2D system is considered to be just an arbitrary external system, then the set of output, Θ_{output}, could be considered to be the set of nine values, $\{a, b, \ldots, i\}$, and may be indexed and structured in any manner that is appropriate within Θ. g_{in} would map these nine values to the specific 54 cells in the 3D cellular automaton using the network index set, D. The output, for instance, may be a (CA-index, value) pair. f_{in} would then have the specific knowledge of the network structure to provide the data to the cells specified by the index set. With this approach, the external mapping functions, g_{in} and g_{out}, and, therefore, the external system, only needs knowledge of how the network references its individual cells and not the structure of the network itself. In the case that the poly-formalism approach is used, the g_{in} and g_{out} functions are speci-

fied within the interaction model. With this proposed approach, the CA formalism may be realized in the GRASS environment.

A general algorithm for executing the composable CA specification is given as:

1. Retrieve the next time interval, h_m.
2. Execute the output function, λ_{ijk}, of each component, M_{ijk}, in the network.
3. Execute $g_{out} \circ f_{out}$ to generate the external output from the network.
4. Execute $f_{in} \circ g_{in}$ to generate external input to each component, M_{ijk}.
5. Execute the transition function, δ_{ijk}, of each component, M_{ijk}.
6. Increment the current time based upon the time interval, h_m.

3 Realization

As stated in Section 1, a CA has potential benefits as a model for many different systems. The *realization* of each (how the formalism is expressed in terms of software architecture and specific domain constraints) depends on the application domain. While this formalism could be used for stand-alone CA systems, its major benefits are gained in a hybrid system. As such, an agent-landscape environment, devised from the Mediterranean Landscape Dynamics (MEDLAND) project [8], will be used as the exemplar domain. MEDLAND is an on-going international and multi-disciplinary effort. One of the project's goals is to develop a laboratory in which to study humans, the environment, and their relationship. In the simulation laboratory, humans are represented by agents and the environment is represented by a landscape model. The landscape model will be the candidate sub-system to demonstrate a realization of the composable CA formalism. Furthermore, an interaction model will be devised to interact with the CA, providing it with input and capturing output.

The interaction model (IM) composes the agent and the landscape sub-system models. It is within the realization of each sub-system model that the interactions between them occur through data transformation, synchronization, concurrency, and timing. The IM is, in essence, a model of the interaction between the two sub-systems models which has its own formalism and realization separate from the two composed sub-system models. It is through the explicit modeling of the interactions within the IM that interaction visibility and, therefore, management of interaction complexity, is gained [5,6]. The exemplar landscape model will realize this CA formalism in the GRASS environment. As a Geographic Information System (GIS), GRASS is described as a georeferenced software development environment that uses map algebra operations to modify its data. Data is stored in a file called a map in either a raster or vector format. GRASS is comprised of independent modules that can acts upon the maps. The modeler is able to define complex dynamics by executing one or more modules against one or maps either manually or via the use of scripts which contain the predetermined execution sequence. From a formalism perspective, map algebra is too general to provide an effective formal representation

of the landscape dynamics that we wish to model. However, the CA specification provided above can be implemented within the GRASS environment by being particular about which GRASS modules are implemented and how they are applied against the mapsets. There is a close resemblance to the regular grid-like tessellation of cells within a GRASS raster map and those within a cellular automaton and it is important to understand how the two data structures relate to one another.

A GRASS raster map may be formatted for either 2-dimensional or 3-dimensional data storage. Each raster map represents one data value (*e.g.*, soil depth). Another value, such as numeric value for land cover, would be stored in a different map. The benefit of using a georeferenced environment like GRASS is that the two maps (assuming they represent the same region) would have the same dimensions, cell resolution, and, therefore, number of cells. Thus, if a cell (x, y) referenced in one would have a representation in the other. Furthermore, GRASS modules are devised such they can perform calculations across maps given these geospatial relationships. In contrast, a cell within a CA has a state. That state can be comprised of many variables (*e.g.*, soil depth **and** land cover). Thus, if a CA is realized within GRASS, the relationship between a cellular automaton cell and GRASS maps is 1-to-many. Also, while a cell may have dimension as part of its state, these values have little impact on the CA as a system. So, to ease the burden of association between the two, it would be beneficial to choose a GRASS region and resolution setting that creates a 1:1 cell count ratio between the GRASS map and the CA index set. Otherwise, a more complex mapping will be required[3].

The exemplar landscape model is a representation of a large watershed area in the early Bronze Age on which farmers tend fields. It is part of a larger system, a hybrid model, which is a composition of a the landscape sub-system model and a farmer sub-system model. The landscape model is itself a composition of three models — land cover, soil erosion, and rainfall[4]. As shown in Fig. 2, these models are interdependent. Soil erosion uses land cover, elevation, and rainfall values to determine how much soil is removed and or deposited in areas. The land cover prevents soil erosion. The heavier the land cover foliage, the lower the erosion in that area. In turn, healthy soil values support larger land cover. If soil is washed away, a smaller amount of foliage is supported. The rainfall model provides the underlying cause of the landscape dynamics. Heavy rains help the land cover grow but also wash away more soil. For simplicity, if the system is modeled over a shorter term, one can assume that the rainfall is constant. Overall, the system operates independently of the farmer model. However, when they are composed using the interaction model, each creates impacts on the other. The farmer, in an effort to create land that is suitable for agriculture, removes some of the land cover. This, in turn, increases the erosion in the area. The reduced soil, combined with the continual farming in the area does not allow the foliage to grow back and soil conditions worsen until farming can no longer be sustained. And, so, the farmer moves to different locations in an effort to

[3] The devising of such a mapping is left to the reader's own discretion.

[4] A discussion on the research for the landscape model may be found in [2, 11, 12].

grow enough food to survive. Land cover may grow back but may not be capable of returning to the condition it once was due to changes in soil quality[5].

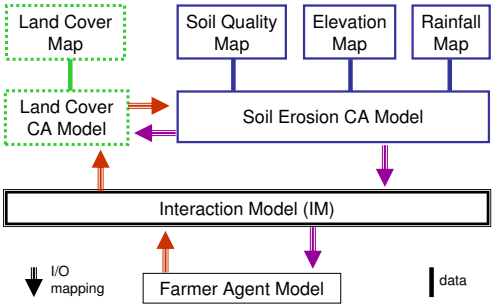


Fig. 2 Exemplar hybrid model.

The dynamics of the CA models can be traced using the algorithm above. For simplicity, we will first assume that each time interval, h_m, has a value of 1. At time t_0, the land cover and soil erosion CA models are initialized. We begin with the first step of the algorithm, retrieving the next time interval, h_0, which is 1. The output function of each CA, λ_{ijk}, then provides output, Y_{ijk}, based upon the initial state of each cell in the respective CA. This occurs both internally as $\dot{Y}_{ijk}$ for every cell that that each influences and externally to create $\overline{Y}_{ijk}$. In step 3 of the algorithm, $\overline{Y}_{ijk}$ is mapped to the input of the receiving system. In the case where the receiving system is another CA, this would be X_N. Step 4 then maps the external input, X_N, to the respective cells, $\overline{X}_{ijk}$, for each CA. This completes the output portion of the execution algorithm. In step 5, each cell in the cellular automata undergoes a state transition based upon the new input values from their influencers, $\dot{X}_{ijk}$, and from external sources, $\overline{X}_{ijk}$[6]. Once the input is accounted for, the cell changes state, as appropriate. The current time of the simulation is then updated by the current time interval, 1, and the algorithm then repeats for time t_1.

The complexity of hybrid model interactions needs to be managed. Even a model as seemingly simple as the one shown in Fig. 2 can be quite complex in its interactions. Each of the land cover, soil erosion, and farmer models may behave in a predictable fashion when tested individually. However, when allowed to interact, the three may exhibit *emergent behaviors* which are not specifically defined in any of the models. To validate the model as a whole, it must be understood why these behaviors are emerging and if they are correct for the inputs being provided across models. This is a difficult task which is made easier through the explicit interaction visibility provided by the interaction model. However, this visibility would be impractical if it were not for a formal composition of the two composed models [6].

[5] The specifics of the hybrid model are not the focus of this paper. The reader is encouraged to read [5, 7] for more details about the models themselves.

[6] The specifics of things such as priority of internal versus external input and managing receipt of multiple external input is a domain specific and is not discussed here.

4 Conclusion

This paper shows key benefits for describing, in a formal way, the methodology by which the composable CA may receive external input and provide output. Developing a CA to this formalism allows the development of a hybrid model which incorporates a cellular automaton, while allowing a rigorous description of the interaction between the CA and the other composed model. It is through formal and explicit modeling of the interactions within the IM that interaction visibility and, therefore, management of interaction complexity, is gained.

Acknowledgements This research is supported by NSF grant #BCS-0140269. We would like to thank the MEDLAND team for their help and partnership; particularly M. Barton and I. Ullah.

References

1. Anderson JA (2006) Automata Theory with Modern Applications. Cambridge University Press, New York.
2. Arrowsmith R, DiMaggio E, et al (2006) Geomorphic mapping and paleoterrain generation for use in modeling Holocene (8,000 - 1,500 yr) agropastoral landuse and landscape interactions in southeast Spain. In: Am Geophys Union, San Francisco, CA.
3. GRASS (2007) Geographic Resources Analysis Support System. http://grass.itc.it/. Cited Nov 2007
4. Ilachinski A (2001) Cellular Automata: A Discrete Universe. World Scientific, New Jersey.
5. Mayer GR, Sarjoughian HS (2007) Complexities of simulating a hybrid agent-landscape model using multi-formalism composability. In: Proc of the 2007 Spring Simulation Conf (SpringSim '07), 161-168, Norfolk, VA.
6. Mayer GR, Sarjoughian HS (2007) Complexities of modeling a hybrid agent-landscape system using poly-formalism composition. In: J Adapt Complex Syst *(submitted)*.
7. Mayer GR, Sarjoughian HS, et al (2006) Simulation modeling for human community and agricultural landuse. In: Proc of the 2006 Spring Simulation Conf (SpringSim '06), 65-72, Huntsville, AL.
8. MEDLAND (2007) Landuse and Landscape Socioecology in the Mediterranean Basin: A Natural Laboratory for the Study of the Long-Term Interaction of Human and Natural Systems. http://www.asu.edu/clas/shesc/projects/medland/. Cited 01 Dec 2007.
9. Ntaimo L, Khargharia B, et al (2004) Forest fire spread and suppression in DEVS. In: Simulation Trans, 80(10), 479-500.
10. Sarjoughian HS (2006) Model composability. In: Perrone LF et al (eds) Proc of the 2006 Winter Simulation Conf (WinterSim '06), 149-158, Monterey, CA.
11. Soto M, Fall P, et al (2007) Land Cover Change in the Southern Levant: 1973 to 2007. Paper presented at the ASPRS Southwest Tech Conf, Arizona State University, Phoenix, AZ.
12. Ullah I, Barton M (2007) Alternative futures of the past: modeling Neolithic landuse and its consequences in the ancient Mediterranean. In: Proc of 72nd Annu Meet of the Soc for Am Archaeol, Austin, TX.
13. Zeigler BP, Praehofer H, Kim TG (2000) Theory of Modeling and Simulation: Integrated Discrete Event and Continuous Complex Dynamic Systems, 2nd edn. Academic Press, CA.