Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

FORMAL SPECIFICATION SUPPORTING INCREMENTAL AND FLEXIBLE AGENT-BASED MODELING

Jang Won Bae
GeunHo Lee
Il-Chull Moon

Department of Industrial and Systems Engineering
KAIST, 291 Daehak-ro
Yuseong-gu, Daejeon 305-701, Republic of Korea

ABSTRACT

Agent-based models have been used for diverse domains such as military, sociology, and urban planning. There is a growing concern about the incrementality and the flexibility of the agent-based models in further sophisticated and large-scale utilization. To resolve this concern, we suggest that specifying agent-based models formally will resolve the problems of incrementality and flexibility of the agent-based models through an organized composition of model components. To organize the composition of agent-based models, we survey formalisms that are applicable to agent-based models, including formalisms from the discrete event models, i.e., DEVS, MDEVS, and Cell-DEVS, as well as formalisms used in the communities of agent-based models, i.e., BDI, MDP, and Game Theory. Then, we compare, contrast, and propose an overarching formal specification for agent-based models that embody the key nature of agents. As an example, we show how to incrementally merge and flexibly manage traditional agent-based models through proposed formal specifications.

1 INTRODUCTION

Agent-based modeling and simulation has been a successful approach in providing insights and predicting the future in sociology (Sakoda 1971), biology (Auyang 1999), management (Robert 1999), military (mittal et al 2007), urban-growth (Benenson 1998), logistics (Barbuceanu, Teigen, and Fox 1997), etc. This success has led to the development of a large number of agent-based models (Heath, Hill, and Ciarallo 2009). For further development and utilization of agent-based models, now the modelers should be able to incrementally build their agent-based models by depending on other modelers' models (Moon and Carley 2007). In the 1970s and 1980s, simple models, such as Schelling's segregation (Schelling 1971), Epstein and Axtell's sugarscape (Epstein and Axtell 1996), and Carley's construct (Carley 1991), provided profound insights into our society. However, to show today's complex socio-economic systems, we need more modeling features that go over more than the coverage of a single model or a single modeler. Furthermore, the sophistication of the modern agent-based models requires more systematic modeling and implementation approaches, while in the past the algorithm, the pseudo-codes and even the source codes of the traditional agent-based models are published in papers. Such model description is now done by utilizing flowcharts, UML notations, or simple textual descriptions. To create more comprehensive models for today's applications, such as a city-scale epidemic model considering the traffic and the social network effects, modelers from different discipline should collaborate and build their models on top of the others'. This collaboration includes not only building incrementally, but also testing and modifying the incrementally built models by changing the incremental composition of its dynamic execution. Enabling such collaboration requires a common ground for their representation, modeling, implementation, and

978-1-4673-4781-5/12/$31.00 ©2012 IEEE

model execution management. We propose that the common ground for modeling is a formal specification for agent-based models.

This paper discusses how to incrementally and flexibly develop and integrate agent-based models through the formal specifications that we propose in this paper. The substance of our approach is specifying a multiple agent-based model in our proposed formal specification in a distributed way that means two modelers are individually developing the models by the same formal specifications. Then, we incrementally compose the specified multiple agent-based models to execute them in the same simulation execution context. Also, we flexibly modify the composition of the multiple models to add and remove some of the models. The incremental composition and the flexible management are enabled by the formal specification of the models because the formal specifications become the protocol in developing, running, and managing the models.

As we point out the importance of the formal specification, to our knowledge, there is no single dominant formalism in the agent-based modeling community. So far, some formalisms specifically designed for agent-based models have been proposed, which will be compared to our approach in the discussion section, yet theses formalisms are not widely used. Therefore, our underlying goal for the above incremental and flexible modeling and simulation is proposing another, more usable formalism for the agent-based modeling community. The existing formalisms for agent-based models can be categorized into two types. First, there are formalisms in some theoretic domains of the agent-based models. For instance, partially observable markov decision-process, or POMDP (Bertsekas 1976), has its own distinct formalism though POMDP is limited in being utilized, i.e., modeling robotic behaviors in a simple environment. Second, there are formalisms reduced from a more general modeling and simulation theory, such as the theory of discrete event system specification, or DEVS (Zeigler, Praehofer, and Kim 2000). For example, DEVS community has suggested a number of formalisms, such as MDEVS (Kim and Kim. 2000), yet these are not popularly used by the agent-based modeling community.

This paper aims to present a third type of formal specification that is adapted more to the agent-based models with its root in DEVS formalism, and we expect more facilitated adoption of it by the agent-based modelers. We present the formal specification in terms of how the specification inherits two cores of the agent-based models and DEVS. We subsequently show a demonstration of our formal specification by applying the specifications to two well-known tradition agent-based models, and then we illustrate how to incrementally compose the fusion model by inheriting the specification of the two models.

2 PREVIOUS RESEARCH

For supporting incrementality and flexibility in the model composition, the formal specifications of models are critical. Since the formal specifications rely on a strict specification basis, such as mathematics, the formal specification becomes the common, comprehensive and accurate protocol in model representing and implementation. Also, if the formal specifications support modular and hierarchical compositions, the specified models are easier to incrementally and flexibly compose as an over-arching model. Then, the question is whether or not the existing formalisms in the agent-based modeling community are 1) comprehensive in modeling agents and 2) hierarchical and modular in composing more than two agent models.

2.1 Existing Formalisms from Discrete Event Models and Agent Models

Agent-based models describe that individual agents interact with other agents and environment. In detail, the agent-based models have three main components: agents, environment, and interactions between them. We surveyed existing formalisms by categorizing the elements of the formalisms into the three categories of the components. This categorization becomes one axis of our survey on formalisms. The second axis of the survey is the origins of the formalisms. Some formalisms adapted to the agent-based models are directly inherited from the cellular automata from the Von Neumann era (Neumann 1966). Another set of formalisms are specialized formalisms from the general discrete event modeling formalism, or DEVS. Finally, some formalisms originated from the AI field to model the details of the agents, such as POMDP,

Table 1 : Surveyed formalisms in agent-based modeling, the words in bold and italics are from the general taxonomy of agent-based models

Categories of tuples in formalisms		Cellular Automata Formalism	DEVS oriented formalisms			Formalisms used for agent modeling in AI		
			DEVS formalism	Mobile-DEVS formalism	Cell-DEVS formalism	BDI based Formal representation	POMDP based formalism	Game theory Formal representation
Agent	Input, or ***Perception***	Set of states of neighborhood	Set of input events	Set of Input events Set of structure-change events	Set of input events Values of input events	Observation	Observation	Action set of other agents
	Output, or ***Action***	State of automaton	Set of output events	Set of output events	Set of output event	Set of action	Set of action	Action of agents
	State	Set of states of automaton	Set of states	Set of states	State variables and values Delay Neighborhood size	Set of belief states	Set of states	
	State Transition, or ***Decision***	Automaton's state transition function	External transition function , Internal transition function	External transition function , Internal transition function	External transition function , Internal transition function , Local computation function ,	Set of inference rules	Set of conditional transition probability , Set of conditional observation probability	Equilibrium algorithm
	Output Function, or ***Behavior***		Output function, Time advance function	Output function, Time advance function	Output function, Transport delay, State's duration function	Plan generator function	Reward function	Pay-off matrix
	Objective					Desire	Maximizing the utility from the reward function	Maximizing the utility in the pay-off matrix
Environment		Interconnection graph			Cell space			
Multi-Agent Coupling Structure, or ***Interactions***	Component, or ***Population***	Set of automatons	Set of component models	Activated models set Model activation function ,	Neighborhood set and size			Set of agents participating games
	Coupling Relation, ***Neighborhood, or Boundary***	Neighborhood definition	External input coupling relation, External output coupling relation, Internal coupling relation	Structure-state set Structure transition function External input coupling relation, External output coupling relation, Internal coupling relation, Change coupling relation	List of input coupling, List of output coupling Modular model interface Translation function Set of border cell			

BDI (Bratman 1987), and Game Theory (Neumann and Morgenstern 1944) .

Table 1 is the result of our survey and is organized with the row axis for key components of agent-based models and the column axis for the origins of the formalism. From the table 1, we recognize that there is no single formalism that comprehensively covers modeling features of agents, environment, and interactions. For instance, Cell-DEVS (Wainer 1998) is one of the most comprehensive formalisms, but it lacks the objective component, such as the desire in BDI agents. Furthermore, we suspect that a more significant barrier in adopting DEVS-oriented formalisms in the general agent-based modeling community is the difficulty in matching and mapping DEVS formalism elements to the features in the agent-based modeling field. For instance, the agent model should autonomously perceive, decide, and act upon the outside stimuli. DEVS oriented formalisms model these critical components, such as perception, action, and decision-making; as input events, output events, and state transition functions. However, this mapping is not well communicated among the agent-based modelers who do not have knowledge of DEVS. In spite of the difficulties in mapping between DEVS and agent-based model, there are many researches solving agent-based models using DEVS. For example, (Sarjoughian, Zeigler and Hall 2001) adopted a layered architecture which is used in DEVS community to agent-based models, (Akplogan et al. 2010) developed agent-based models in agriculture based on DEVS and (Zhang, Chan and Ukkusuri 2011) developed transportation evacuation models using hybrid simulation based on DEVS. Furthermore, there is a recent research about formal semantics of multi-agent simulations based on DEVS formalism, called M-DEVS (Müller 2009).

On the other hand, the formalisms in modeling the agent behavior are often limited to the detailed description of the agent itself. Table 1 shows that the formalisms from the AI have a limited collection of environment- and interaction-related features. Furthermore, the formalisms model one action of an agent, which is just one facet of multiple actions that the agent might exhibit. Also, these formalisms are often used for developing prescriptive models that generate behavior through inferences, rather than descriptive models that have fixed behavior details. Therefore, modelers using simple rules or detailed descriptions of behavior may not be able to use the AI oriented formalisms for the agent models. Having said this, the formalisms from AI are more adaptable to the agent-based models in terms of the taxonomy wording of the formalisms. They name the elements as actions, observations, utility, etc., which are naturally understood by the agent-based modelers unlike DEVS's taxonomy. Formal specification

3 FORMAL SPECIFICATION

This section illustrates 1) how agent-based models are structured in general and 2) key elements, which are actions, agents, and multi-agents, in our formal specification.

3.1 Hierarchical Structure of Agent-Based Model

Before introducing the formal specification for agent-based models, we illustrate a general structure of agent-based models. Traditionally, the big picture of agent-based models is described as a diagram(Russel and Norvig 1995) in Figure 1.

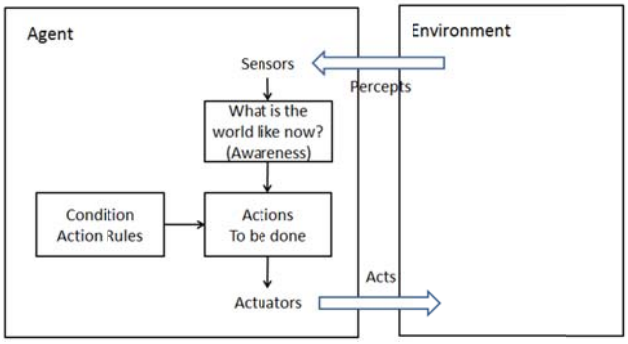

Figure 1: Simplex agent and environment

The diagram shows how agents and environments interact with each other, and the simplified internal architecture of agents. As this diagrams capture the key elements, i.e., agents, perceptions, actions, environments, etc., of the agent-based models, our specification should be accommodated to specify the captured elements.

From the perspective of model composition, Figure 2 shows the hierarchy of the general structure of the agent-based models. Generally, the agent-based models have agents and environments. The interactions between *agent models* and between an *agent model* and an *environment model* are enabled through the composition of agents and environment by the *multi-agents models* and the *agent-based model*, respectively, in Figure 2. This means that the *agent-based model* takes a role of a collective model propagating interactions between the *multi-agents model* and the *environment*. The *multi-agents model* contains multiple *agent models*, and the *multi-agents model* becomes the broker to transfer interactions between *agent models*. This composition and role goes same to the *environment model*. The *agent model* can perform multiple actions as modeled in *action models*. We limited our description on the environment models because this paper is limited to introducing only the action, the agent, and the multi-agents models' specifications.

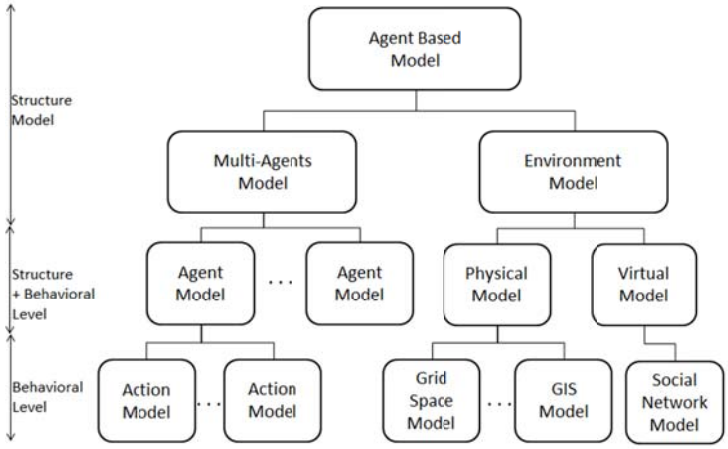

Figure 2: Hierarchy of the general structure of agent-based models

3.2 Formal Specification of Action Model

The *action model* in Figure 2 describes an agent's behavior in a single facet under the assumption that agents may exhibit multiple actions in different domains. For example, let's imagine. An ambulance as a car in the traffic and as a rescue resource in the emergency medicine. The ambulance may exhibit the maneuver action and the health-care action, simultaneously. This *action model* describes a single action out of two facets of the agent's action. In order to specify an *action model*, we consider the general procedures of the agent's actions. According to agent's skeleton in Figure 1, the action of an agent can be expressed into three stages: the perception stage, the decision stage and the action stage.

- Perception stage: Agents recognize the external information from environments or other agents
- Decision stage: Agents decide his next actions as response
- Action stage: Agents affect the environments or other agents with their actions

Such procedures of action take parameters from the state of *action model*. In order to specify the states, we split the states into three types : situation awareness, action, and condition. This separation of states is one of the distinct aspects of our formal specification. As we separate the actions into three stages, some states are exclusively used for a certain action stage, i.e., situation awareness in perceiving the stimuli of agents. Therefore, we separated the states into three sets to further specifically describe the *action model* rather than having a single state set.

- States of situation awareness: States from the external information from environments and other agents
- States of action: States of executing a certain action
- States of condition: States describing condition-action rules in the decision-making process

The above characterization of actions results in the formal specifications for *action model* below. An *action model* is formally specified by five sets and four functions. Below, X means an input event set from external models, such as other agents and environments. Y means an output event set which is generated by the *action model*. The states of the *action model* are divided into three types : states of situation awareness(S_{aw}), states of actions(S_{act}), and states of conditions(S_{cond}). Using these state sets, four functions need to be specified, and the four functions include three basic functions from the agent's skeleton in Figure 1 and a time advance function that is essential in discrete event system. The perceive function (P) means, when the external events come in, an *action model* changes its situation awareness states by the external inputs. The decision function (D) specifies the transition of its action states according to its situation awareness, action in progress, and decision conditions. The action function (A) describes an *action model* generating an output event from the action states. Lastly, the time advance function defines the time of the action completion by the chosen output action. While the mathematical notations are described below, Figure 3 shows the structure of the *action model* in the formal specification .

$ACT = <X, Y, S_{aw}, S_{act}, S_{cond}, P, D, A, ta>$
X = Set of input event
Y = Set of output event
S_{aw} = Set of states about situation awareness
S_{act} = Set of states about action
S_{cond} = Set of states about the agent's conditions
$P: (X \times S_{aw}) \rightarrow S_{aw} \times S_{cond}$ = Perceive function
$D: (S_{aw} \times S_{act} \times S_{cond}) \rightarrow S_{act}$ = Decide function
$A: (S_{act}) \rightarrow Y$ = Action Function
$ta: (S_{act}) \rightarrow R^+$ = Time Advance Function

3.3 Formal Specification of Agent Model

The *agent model* in Figure 2 specifies an agent entity performing multiple actions specified in its action models. When the model is executed, the *agent model* performs the actions dynamically composed with respect to the *agent model's* states and external events generated by either other agents or environments. This dynamic action composition is enabled by the coupling structure between the action models within the *agent model*. To express this action composition in an agent, the following is a formal specification of an *agent model*

$AM = < X, Y, S, A, \delta, C, SELECT >$
X = Set of input event
Y = Set of output event
S = Set of states about coupling structure
A = Set of action models
$\delta: (X \times S) \rightarrow S$ = State Transition Function
$C = \{sEIC, sEOC, sIC\}$, Coupling Structure set
$sEIC \subseteq S \times X \times \cup X_i$, where $X_i \in$ X of A, state based External Input Coupling
$sEOC \subseteq S \times X \times \cup X_i$, where $X_i \in$ X of A, state based External Output Coupling
$sIC \subseteq S \times \cup Y_i \times \cup X_i$, where $X_i \in$ X of A and $Y_i \in$ Y of A, state based Internal Coupling
$SELECT: 2^{\{A\}} - \emptyset \rightarrow A$, Select Function,

An *agent model* consists of seven tuples. *X* and *Y* mean the sets of the inputs and the output events of the *agent model*, respectively. *S* means the set of the states about coupling structure. *A* means the set of actions in *the agent model*. *C* describes coupling relations between an *agent model* and its action models. The coupling relations consist of *sEIC*, *sEOC,* and *sIC*. *sEIC* describes relations between an input event of the agent model and an input event of one of its action models. *sEOC* is similar to *sEIC* while *sEOC* handles the outputs. *sIC* describes relations between an input event and an output event of action models in the *agent model*. The state transition function (δ) determines the composition state of the actions of the *agent model*. The select function (SELECT) determines the priority of executing an action when multiple actions are required to be executed by an input. This select function is adapted from DEVS formalism with minimal change, so further details is in (Zeigler, Praehofer, and Kim 2000).

3.4 Formal Specification of Multi-Agents Model

The *multi-agents model* in Figure 2 describes interactions between agents. The *multi-agents model* includes multiple agent models with a coupling structure. This coupling structure might be changed by either output of agent models (i.e. the movement of an ambulance) or input events from environment models. The coupling structure at a certain moment is determined by the coupling structure state. If we substitute 1) the agent models with the action models and 2) the *multi-agents model* with the agent models, the composition of the *multi-agents model* is same to the composition of the agent model in the previous section except one difference. The difference is that the *multi-agents model* changes its coupling structure by the output events generated by its agent models and the input events from external models, while the agent model changes its coupling structure by input events to the agent model itself. This difference is annotated in Figure 3.

$MAM = < X, Y, S, AM, C, \delta, SELECT>$
$X = \{X_{cs} \cup X_{in}\}$, where X_{cs} = Set of structure change events and X_{in} = Set of input events
Y = Set of output events
S = Set of states about coupling structure
AM = Set of agent models
$\delta = X_{cs} \times S \rightarrow S$, Coupling Structure transition function
$C = \{sEIC, sEOC, sIC, sCS\}$, Coupling Structure set
$sEIC \subseteq S \times X \times \cup X_i$, where $X_i \in$ X of AM, state based External Input Coupling
$sEOC \subseteq S \times X \times \cup X_i$, where $X_i \in$ X of AM, state based External Output Coupling
$sIC \subseteq S \times \cup Y_i \times \cup X_i$, where $X_i \in$ X of AM and $Y_i \in$ Y of AM, state based Internal Coupling
$sCS \subseteq S \times \cup Y_i \times X_{cs}$, where $Y_i \in$ Y of AM, Coupling Structure Relation
$SELECT : 2^{\{AM\}} - \emptyset \rightarrow AM$, Select Function,

To explain the formal specification of the *multi-agents* model, we only explain the different elements compared to the formal specification of the agent model. Coupling structure relation (*sCS*) is added to link the outputs of the agent model to the *multi-agents model*'s state-changing inputs (X_{cs}). Following figure 3 show the structures of the *action*, the *agent* and the *multi-agents model* specifications.

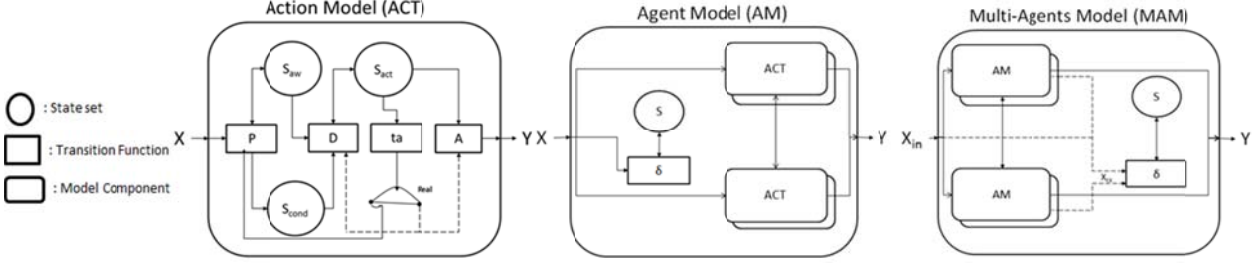

Figure 3: Structure of the action model (left), agent model (mid) and multi-agents model (right)

3.5 Reduction to DEVS Formalism

In this section, we prove that our formal specification is reducible to DEVS formalism. The reducibility to DEVS will show that the suggested formal specification is sufficient for modeling in the hierarchical and modular fashion, and our formal specification belongs to a special case of DEVS formalism. Table 2 shows that our formal specification has matching and more elements to DEVS formalism. This means that our formal specification is reducible to DEVS formalism by removing some information in our specification. In detail, the action model is reducible to the atomic model in DEVS. These two models are same except our specification splits the state set of DEVS into three sets in our model. The agent model is reducible to the coupled model in DEVS while our model has its own state set and transition function determining the coupled structure. Having the state set and the transition function is first introduced by the MDEVS formalism (Kim and Kim 2000) that we surveyed before, yet we simplified and reorganized the formalism to match the agent-based modeling context. The multi-agents model is another extension of DEVS coupled model, and the result of the reduction is the same as the agent model's reduction.

Table 2: Reduction from the proposed formal specification to DEVS formalism

	DEVS Formalism		Formal Specification for Agent-Based Models		
	Atomic Model	Coupled Model	Action Model	Agent Model	Multi-Agents Model
Input Event	X	X	X	X	$X_{cs} \cup X_{in}$
Output Event	Y	Y	Y	Y	Y
States	S		$S_{aw} \times S_{act} \times S_{cond}$	S	S
External Transition Function	$\delta_{ext} : (X \times S) \rightarrow S$		$P : (X \times S_{aw}) \rightarrow S_{aw} \times S_{cond}$	$\delta = (X \times S) \rightarrow S,$	$\delta = (X_{cs} \times S) \rightarrow S$
Internal Transition Function	$\delta_{int} : S \rightarrow S$		$D : (S_{aw} \times S_{act} \times S_{cond}) \rightarrow S_{act}$		
Output Function	$\lambda : S \rightarrow Y$		$A : S_{act} \rightarrow Y$		
Time Advance Function	$ta : S \rightarrow R^+$		$ta : S_{act} \rightarrow R^+$		
Component Model		M		A	AM
Coupling Structure		$\{EIC, IC, EOC\}$		$\{sEIC, sEOC, sIC\}$	$\{sEIC, sEOC, sIC, CS\}$
SELECT Function		$SELECT$		$SELECT$	$SELECT$

3.6 Example : Construct-Spatial Model

We give an example to demonstrate the formal specification of agent-based models. Whereas there are few agent-based models that incrementally and flexibly integrates existing models, some integration models (Moon and Carley 2007) are created by manually integrating two existing models without any support from formalisms. Construct-spatial is one attempt to create a mixture of two agent-based models in the spatial and the network domains. Construct-spatial model is an agent-based model of mixing 1) the action of the sugarscape agent (Epstein and Axtell 1996) that moves in a physical environment and 2) the action of the construct agent (Carley and Hill 2001) that communicates on a social network environment.

- Sugarscape Action (SA) : an action about moving toward resources (i.e. sugar)
- Construct Action (CA) : an action about communicating knowledge with other agents
- Construct-Sugarscape Agent (CSA) : an agent who can perform one of SA and CA on the specific condition.

3.6.1 Action Model : Sugarscape Action Model

The sugarscape action model describes moving toward resources (i.e. sugar). This action model gets the location of resources within their vision and generates their updated position. If the action model finds the location of resource, it goes toward the resource. Otherwise, it stays. Although the action model finds the

resource, it may not move by the condition set. the formal specification of the sugarscape action model is below.

Sugarscape Action Model (SA) = < X, Y, S_{aw}, S_{act}, C, P, D, A, ta>
X = { $Grid_{res}$, $Agent_{res}$ | $Grid_{res}$ = (x_{grid}, y_{grid}, s_{grid}), x_{grid} = x coord. of grid environment in vision range,
y_{grid} = y coord. of grid environment in vision range, s_{grid} = amount of sugar distribution ,
$Agent_{res}$ = (x_i, y_i, s_i, a_i), x_i, y_i and s_i are same definitions in $Grid_{res}$,
a_i = index of agent who sends the information of sugar to agent including CA }
Y = { Loc_{agent} | Loc_{agent} = (x_{agent}, y_{agent}, a_{agent})
, x_{agent}, y_{agent} =x, y coord. of grid environment where agent is located, a_{agent} = index of the agent including SA }
S_{aw} = { $Target_{res}$= (x_{tar}, y_{tar}, s_{tar}) | x_{tar}, y_{tar}, and s_{tar} use the same definitions in X }
S_{act} = { (movement_action) | movement_action=either GoResource(x,y (x,y coord. of resource)) or Stay}
S_{cond} = { (x_{loc},y_{loc}, health) | x_{loc}, y_{loc} =x, y coord. of the agent location, health=either hungry or full}
P : (X × S_{aw}× S_{cond}) → S'_{aw} × S'_{cond}
(x_{grid}, y_{grid}, s_{grid}) ×(x_{tar}, y_{tar}, s_{tar})×(x_{loc}, y_{loc}, Full) → (x_{tar}, y_{tar}, s_{tar})×Hungry
(x_{grid}, y_{grid}, s_{grid}) ×(x_{tar}, y_{tar}, s_{tar})×(x_{loc}, y_{loc}, Hungry) → (x_{grid}, y_{grid}, s_{grid})×Hungry , if (s_{grid} > s_{sa})
(x_{grid}, y_{grid}, s_{grid}) ×(x_{tar}, y_{tar}, s_{tar})×(x_{loc}, y_{loc}, Hungry) → (x_{sa}, y_{sa}, s_{sa})×Hungry , if (s_{grid} <= s_{sa})
(x_i, y_i, s_i, a_i) ×(x_{tar}, y_{tar}, s_{tar})×(x_{loc}, y_{loc}, Full) → (x_{tar}, y_{tar}, s_{tar})×Hungry
(x_i, y_i, s_i, a_i) ×(x_{tar}, y_{tar}, s_{tar})×(x_{loc}, y_{loc}, Hungry) → (x_{grid}, y_{grid}, s_{grid})×Hungry , if (s_i > s_{sa})
(x_i, y_i, s_i, a_i) ×(x_{tar}, y_{tar}, s_{tar})×(x_{loc}, y_{loc}, Hungry) → (x_{sa}, y_{sa}, s_{sa})×Hungry , if (s_i <= s_{sa})
D : (S_{aw} × S_{act} × S_{cond}) → S'_{act}
(x_{tar}, y_{tar}, s_{tar})×S_{act}×(x_{loc}, y_{loc}, Full) → Stay
(x_{tar}, y_{tar}, s_{tar})×S_{act}×(x_{loc}, y_{loc}, Hungry) → GoResource(x_{tar}, y_{tar})
A : S_{act} → Y
Go_Resource(x,y)→ (x',y')
Stay→ (x,y)
ta : S_{act} → R^+
Go_Resource(9,10) → t_{go} ∈ R^+
Stay → ∞

3.6.2 Action Model : Construct Action Model

The construct action model describes communicating their knowledge about resources with other agents. This action model communicates with other agents within vision range. The construct action model has a list of agents within vision range and it can communicate with them. According to the condition set of the action model, the action model decides whether it communicates with other agents or not. Once it decides to communicate, it generates an event to change coupling structure to a multi-agents model which the action model is in. The formal specification of the construct action model is below.

Construct Action Model (CA) = < X, Y, S_{aw}, S_{act}, C, P, D, A, ta>
X = { $Grid_{res}$, $Agent_{res}$, $List_{agent}$| $Grid_{res}$ = (x_{grid}, y_{grid}, s_{grid}), x_{grid} = x coord. of grid environment in vision range,
y_{grid} = y coord. of grid environment in vision range, s_{grid} = amount of sugar distribution
, $Agent_{res}$ = (x_i, y_i, s_i, a_i), x_i, y_i and s_i are same definitions in $Grid_{res}$,
a_i = index of agent who sends the information of sugar to agent including CA
, $List_{agent}$ = List of indexes of agents who are possible to communicate in the vision range }
Y = { ($Agent_{res}$, NetworkChange |
$Agent_{res}$ = (x_{ca},y_{ca},s_{ca},a_{ca}) same definition in X, NetworkChange = (a_{ca}, a_i), a_i = index of $agent_i$ }
S_{aw} = { $Agent_{res}$ × $List_{agent}$ | $Agent_{res}$ = (x_{ca},y_{ca},s_{ca},a_{ca}), x_{ca},y_{ca},s_{ca},a_{ca} are same definition in X
, $List_{agent}$ = index of agents who are in the vision range }
S_{act} = { CommAction | CommAction = either Communicate(x_{ca},y_{ca},s_{ca},a_{ca}, a_{agent}),
x_{ca},y_{ca},s_{ca},a_{ca}, a_{agent} are same definition in S_{aw} or Wait }
S_{cond} = { Characteristic | Characteristic = either Friendly or Selfish }
P : (X × S_{aw}× S_{cond}) → S'_{aw}×S'_{cond}
(x_{grid}, y_{grid}, s_{grid}) × ((x_{ca},y_{ca},s_{ca},a_{ca}) × $List_{agent}$) × S_{cond} → ((x_{grid}, y_{grid}, s_{grid}, a_{ca}) × $List_{agent}$) × S_{cond}

$(x_i, y_i, s_i, a_i) \times (x_{ca},y_{ca},s_{ca},a_{ca}) \times \text{Friendly} \rightarrow ((x_i,y_i,s_i,a_i) \times \text{List}_{agent} \cup \{a_i\}) \times \text{Friendly}$,if $(s_i > s_{ca})$

$(x_i, y_i, s_i, a_i) \times (x_{ca},y_{ca},s_{ca},a_{ca}) \times \text{Friendly} \rightarrow ((x_{ca},y_{ca},s_{ca},a_{ca}) \times \text{List}_{agent} \cup \{a_i\}) \times \text{Selfish}$,if $(s_i <= s_{ca})$

$(\text{List}_{agent}) \times ((x_{ca},y_{ca},s_{ca},a_{ca}) \times a_{agent}) \times \text{Selfish} \rightarrow ((x_{grid}, y_{grid}, s_{grid}, a_{ca}) \times \text{List}_{agent} \times \text{Friendly}$

D : $(S_{aw} \times S_{act} \times S_{cond}) \rightarrow S'_{act}$

$(S_{aw} \times S_{act} \times \text{Selfish} \rightarrow \text{Wait}$

$((x_{ca},y_{ca},s_{ca},a_{ca}) \times \text{List}_{agent}) \times S_{act} \times \text{Friendly} \rightarrow \text{Communicate}(x_{ca},y_{ca},s_{ca},a_{ca}, a_{agent}), a_{agent} \in \text{List}_{agent}$

A : $S_{act} \rightarrow Y$

$\text{Wait} \rightarrow \{ \}$

$\text{Communicate}(x_{ca},y_{ca},s_{ca},a_{ca}, a_{agent}) \rightarrow \text{Networkchange}(a_{ca},a_{agent}), \text{Agent}_{res}(x_{ca},y_{ca},s_{ca},a_{ca})$

ta : $S_{act} \rightarrow R^+$

$\text{Wait} \rightarrow \infty$

$\text{Communicate} \rightarrow t_{comm} \in R^+$

3.6.3 Agent Model : Construct-spatial Agent Model

Construct-spatial agent model is generated by composition of the sugarscape action model and Construct action model. According to the states of the agent model and the external events from other agents or environment models, the agent model decide an action model to be executed at certain time, which is regulated by changing coupling structure of the agent model. The formal specification of Construct-spatial agent model is below.

Construct-Spatial Agent Model (CSA) = < X, Y, S, A, C, δ, SELECT>

X = { Grid_{res} , InAgent_{res}, List_{agent} | same definitions in CA, SA }

Y = { OutAgent_{res}, NetworkChange, Loc_{agent} | same definitions in CA, SA }

S = { Construct, Sugarscape, Both }

A = { CA, SA}

C = {sEIC, sEOC, sIC }, Coupling structure set

sEIC ⊆ {S ×(CSA, Grid_{res}) × (CA, Grid_{res}), S× (CSA, Grid_{res}) × (SA, Grid_{res}),
S × (CSA, InAgent_{res}) × (CA, Agent_{res}), S × (CSA, InAgent_{res}) × (SA, Agent_{res}),
S× (CSA, List_{agent}) × (CA, List_{agent}) }

sEOC ⊆ { S × (CA, Agent_{res}) × (CSA, OutAgent_{res}), S ×(SA, Loc_{agent}) × (CSA, Loc_{agent}),
S× (CA, NetworkChange) × (CSA, NetworkChange) }

sIC ⊆ { S× (CA, Agent_{res}) × (SA, Agent_{res}) }

δ : (Grid_{res} × S) → Both

(InAgent_{res} × S) → Both

(List_{agent} × S) → Construct

SELECT : $2^{\{A\}} - ø \rightarrow A_i, A_j \in A$

3.6.4 Multi-Agents Model : Construct-spatial Multi-Agents Model

Construct-spatial multi-agents model contains multiple developed Construct-spatial agent models and specify the coupling structure of the agent models. on the contrast to Construct-spatial agent models, this multi-agents model changes its coupling structure by the output events from its agent models. The formal specification of the multi-agents model is below.

Construct-spatial Multi-agents Model (CSMA) = < X, Y, AM, S, C, δ, SELECT>

X = {$X_{cs} \cup X_{in}$}, X_{cs} = { NetworkChange }, X_{in} = { Grid_{res} , List_{agent}}, same definitions in CSA model

Y = { Loc_{agent} }, same definitions in CSA model

AM = {M_i | M_i is a CSA model with indexing i }

S = { M_i | $M_i \in$ AM , M_i is an activate agent }

C = {sEIC, sEOC, sIC, CS}, Coupling structure set

sEIC $\subseteq$ (S × (CSMA, $Grid_{res}$) × (M_i, $Grid_{res}$)), (S × (CSMA, $List_{agent}$) × (M_i, $List_{agent}$)), $M_i \in$ AM

sEOC $\subseteq$ (S × (M_i, Loc_{agent}) × (CSMA, Loc_{agent})), $M_i \in$ AM

sIC $\subseteq$ (S × (M_i, $OutAgent_{res}$) × (M_j, $InAgent_{res}$)), M_i and $M_j \in$ AM

CS $\subseteq$ (S × (M_i, $OutAgent_{res}$) × (CSMA, NetworkChange), $M_i \in$ AM

$\delta = X_{cs} \times S \rightarrow S'$

NetworkChange(a_i, a_j) × S → S’ , a_i, $a_j \in$ AM

SELECT : $2^{\{AM\}} - \emptyset \rightarrow M_i, M_j \in$ AM

Figure 4 shows the structure of Construct-spatial agent model and multi-agent model as the formal specifications of the models.

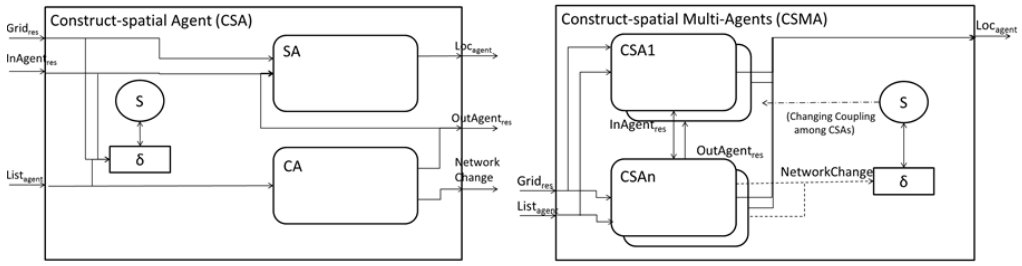

Figure 4: Structure of Construct-spatial agent model (left) and multi-agents model (right)

4 CONCLUSION

This paper proposes a formal specification of agent-based models to support the incremental and the flexible models development. Our formal specification, which is reducible to DEVS formalism, enables the hierarchical and the modular modeling of the action, the agent, and the multi-agents models. Particularly, the action models are expressed by common taxonomy in the agent-based modeling community, i.e., perception, decision, and action. Additionally, we organize a simple, yet sufficient formal specification in the dynamic composition of 1) action models in an agent model and 2) agent models in a multi-agents model. To demonstrate the expressive power of our formal specification, we show an example of construct-spatial model that is composed of two well-known traditional agent-based models: sugarscape and construct. To sustain this formal specification, we plan to provide a model implementation platform supporting this formal specification.

REFERENCES

Akplogan, Mahuna, Gauthier Quesnel, Alexandre Joannon, and Roger Martin-clouaire. 2010. Towards a deliberative agent system based on DEVS formalism for application in agriculture. In Proccedings of the Summer Computer Simulation Conference SCSC10 (2010).

Auyang, Sunny Y.. 1999. *Foundations of Complex-System Theories: In Economics, Evolutionary Biology, and Statistical Physics*. Cambridge University Press.

Barbuceanu, Mihai, Rune Teigen, and Mark S. Fox. 1997. Agent based design and simulation of supply chain systems. In Proceedings of IEEE 6th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 36-41. IEEE Comput. Soc. doi:10.1109/ENABL.1997.630787.

Benenson, tzhak. 1998. “Multi-agent simulations of residential dynamics in the city.” Computhers, Environment and Urban Systems 22 (1): 25-42. doi:10.1016/S0198-9715(98)00017-9. http://www.sciencedirect.com/science/article/pii/S0198971598000179.

Bernard, Robert N.. 1999. *Using Adaptive Agent-Based Simulation Models to Assist Planners in Policy Development: The Case of Rent Control*. SANTA FE. http://www.santafe.edu/research/working-papers/abstract/0cbe9b4e0fc8061589e453dea9f7721e/.

Bertsekas, D. P.. 1976. *Dynamic programming and stochastic control*. New York: Academic. 111-128.

Bratman, Michael E.. 1987. *Intention, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press.

Carley, K. M. 1991. *A Theory of Group Stability,* American Sociological Review 56 (3): 331-354.

Epstein, Joshua M. and R. Axtell. 1996. *Growing artificial societies: social science from the bottom up*. Cambridge MA: MIT Press.

Gaupp, Martin P. and R.R. Hill. 1999. *Using adaptive agents in Java to simulate U.S. air force pilot retention*. In 1999 Winter Simulation Conference (WCS'99) - volume 2. Phoenix, AZ, USA.

Heath, Brian, Raymond Hill, and Frank Ciarallo. 2009. "A Survey of Agent-Based Modeling Practices (January 1998 to July 2008)." Journal of Artificial Societies and Social Simulation 12 (4): 9.

Kim, Jae-Hyun and Tag Gon Kim. 2000. *FRAMEWORK FOR MODELING/SIMULATION OF MOBILE AGENT SYSTEMS*. In Proceedings of 2000 Conference on AI, Simulation and Planning in High Autonomy Systems, 53-59. http://smsl.kaist.ac.kr/paper/CF/CF-63.pdf.

Mittal, Saurabh, Bernard P Zeigler, Jose L Risco, and Jesús M De. 2007. WSDL-BASED DEVS AGENT FOR NET-CENTRIC SYSTEMS ENGINEERING. In DEVS Integrative M&S Symposium DEVS'07.

Moon, Il-Chul and Kathleen M Carley. 2007. Self-Organizing Social and Spatial Networks under What-if Scenarios. In Autonomous Agent and Multi-Agent Systems 2007, 127-134. Honolulu, Hawaii. http://seslab.kaist.ac.kr/xe/321.

Müller, Jean-pierre. 2009. Towards a Formal Semantics of Event-Based Multi-agent Simulations. In 10th International Workshop on Multi-Agent-Based Simulation, 110-126.

Neumann, John von and Oskar Morgenstern. 1944. *Theory of Games and Economic Behavior*. 60th ed. USA: Princeton University Press.

Neumann, John Von. 1966. Theory of Self-Reproducing Automata. Ed. Arthur W. Burks. Urbana and London: University of Illinois Press. http://cba.mit.edu/events/03.11.ASE/docs/VonNeumann.pdf.

RUSSEL, S. and P. Norvig. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall. Upper Saddle River. NJ.

Schelling, T.. 1971. "Dynamic Models of Segregation." Journal of Mathematical Sociology 1:143-86.

Sakoda, J. M.. 1971. *The checkerboard model of social interaction*. J Math Social 1: 119–132.

Sarjoughian, H S, B P Zeigler, and S B Hall. 2001. A layered modeling and simulation architecture for agent-based system development. Proceedings of the IEEE. Vol. 89. IEEE. doi:10.1109/5.910855. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=910855.

Wainer, Gabriel A.. 1998. Discrete events cellular models with explicit delays. Universidad Nacional de Buenos Aires. http://www.emis.ams.org/journals/SADIO/vol.2.1/tesis3.html.

Zhang, Bo, Wai Kin Chan, and Satish V. Ukkusuri. 2011. Agent-based discrete-event hybrid space modeling approach for transportation evacuation simulation. In Proceedings of the 2011 Winter Simulation Conference (WSC), 199-209. http://www.informs-sim.org/wsc11papers/017.pdf.

Zeigler, Bernard P, Herbert Praehofer, and Tag Gon Kim. 2000. Theory of Modeling and Simulation. Simulation. Vol. 132. Academic Press. doi:10.1159/000074301.

AUTHOR BIOGRAPHIES

JANG WON BAE is a doctoral student at Department of Industrial and Systems Engineering, KAIST. His email address is repute82@kaist.ac.kr.

GEUNHO LEE is a master candidate at Department of Industrial and Systems Engineering, KAIST. His email address is geunho@kaist.ac.kr.

IL-CHUL MOON is an assistant professor at Department of Industrial and Systems Engineering, KAIST. His email address is icmoon@kaist.ac.kr.