Formalization of Weak Emergence in Multiagent Systems

CLAUDIA SZABO, School of Computer Science, The University of Adelaide, Adelaide, Australia
YONG MENG TEO, Department of Computer Science, National University of Singapore, Singapore

Emergence becomes a distinguishing system feature as system complexity grows with the number of components, interactions, and connectivities. Examples of emergent behaviors include the flocking of birds, traffic jams, and hubs in social networks, among others. Despite significant research interest in recent years, there is a lack of formal methods to understand, identify, and predict emergent behavior in multiagent systems. Existing approaches either require detailed prior knowledge about emergent behavior or are computationally infeasible. This article introduces a grammar-based approach to formalize and identify the existence and extent of emergence without the need for prior knowledge of emergent properties. Our approach is based on weak (basic) emergence that is both generated and autonomous from the underlying agents. We employ formal grammars to capture agent interactions in the forms of words written on a common tape. Our formalism captures agents of diverse types and open systems. We propose an automated approach for the identification of emergent behavior and show its benefits through theoretical and experimental analysis. We also propose a significant reduction of state-space explosion through the use of our proposed degree of interaction metrics. Our experiments using the boids model show the feasibility of our approach but also highlight future avenues of improvement.

Categories and Subject Descriptors: I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*

General Terms: Emergence, Theory, Verification

Additional Key Words and Phrases: Emergent behavior, complex system, multiagent system, simulation

ACM Reference Format:
Claudia Szabo and Yong Meng Teo. 2015. Formalization of weak emergence in multiagent systems. ACM Trans. Model. Comput. Simul. 26, 1, Article 6 (September 2015), 25 pages.
DOI: http://dx.doi.org/10.1145/2815502

1. INTRODUCTION

Existing systems nowadays have a large number of components that exhibit complex interconnections [Reynolds 1987; Zhan et al. 2008]. As components interact and make decisions individually based on their internal logic, such systems can usually be modeled as multiagent systems, where agents play the role of components [Heath et al. 2009]. A system is said to be complex if it exhibits designed, expected properties that can be derived from the system specification, as well as properties that are irreducible from knowledge of the interconnected components [Darley 1994; Deguet et al. 2006; Li et al. 2006]. These irreducible properties are called *emergent properties* or *emergence*. Flocking is, for example, an emergent property of a group of birds [Reynolds

This work is supported by the National University of Singapore under grant number R-252-000-522-112 and R-252-000-470-112.
Authors' addresses: C. Szabo, School of Computer Science, The University of Adelaide, Adelaide 5005, Australia; email: claudia.szabo@adelaide.edu.au; Y. M. Teo, Department of Computer Science, National University of Singapore, Singapore 117417, Singapore; email: teoym@comp.nus.edu.sg.

DOI: http://dx.doi.org/10.1145/2815502

1987]. Emergent phenomena are abundant in computer systems [Bedau 1997; Chi 2009; Floyd and Jacobson 1994; Holland 1997; Johnson 2006; Mogul 2006; Ramakrishnan and Yang 1994]. For example, the distribution of links in the World Wide Web scales according to a power law in which a few pages are linked to many times and most are seldom linked to [Adamic and Huberman 2000]. A related property of the network of links in the World Wide Web is that almost any pair of pages can be connected to each other through a relatively short chain of links [Albert et al. 1999]. Other examples include patterns in the Game of Life, traffic jams, unwanted synchronization in distributed systems, and router failures in networks [Mogul 2006].

The study of emergence provides great potential to understand agent and environment interactions [Chan 2010; Chen et al. 2009a] and to explain how natural systems work [Seth 2008]. As emergent properties are identified, it is also important to know if they are welcomed in the system or harmful so that we can exploit the positive impacts of the beneficial properties and minimize, or even prevent, the negative consequences of harmful properties. An example of beneficial property is when users adapt software products to support tasks that the designer never intended. Harmful emergent properties such as unwanted synchronization and oscillation in distributed systems may lead to unforeseeable failures and can make a system harder to design, analyze, and control [Mogul 2006]. Thus, the appearance of emergent properties in a system should be predicted and its type (i.e., beneficial or harmful) should be highlighted. Predicting that emergent behavior will occur or, alternatively, identifying emergent behavior as it happens remains a challenging task [Rouff et al. 2004; Dyson 1998; Randles et al. 2007]. According to Dyson [1998], emergent phenomena cannot be predicted through analysis at any level simpler than that of the system as a whole. Although in some cases emergent properties can be regular, that is, recurring and recognizable, this does not imply that they are easily recognized [Holland 1997]. The most frequent studied type of emergence, *weak* (or *basic*) emergence [Bar-Yam 2004; Bedau 2003; Chalmers 2006; Fromm 2007; Kubik 2003], considers that the system properties can be eventually traced back to the properties of the individuals, but that computer-aided analysis is required. This is the simplest form of emergence, but a practical, automated approach for its analysis and identification has yet to be proposed. As such, we focus on identifying this type in this article and propose an approach that can be extended to study *strong* emergence in future work.

Very few methods for the identification, classification, and analysis of emergent properties exist [Chen et al. 2009a; Kubik 2003; Seth 2008; Szabo and Teo 2012a], and existing methods are usually illustrated using simplified examples. These approaches can be classified broadly from two orthogonal perspectives. In the first perspective, approaches propose to identify emergence as it happens [Kubik 2003; Szabo and Teo 2012a] and aim to use formal or meta-models of calculated composed model states. These approaches rely on the identification of variables or attributes that describe the system components, or the *micro-level*, and the system as a whole, or the *macro-level*, and the relationships and dependencies between these two levels. Emergence is then defined as the set difference between the macro-level and the micro-level. However, in systems with a large number of complex entities with prolonged interactions, identifying the micro- and macro-levels is a key issue. Moreover, the calculation of emergence is computationally expensive. In contrast, the second perspective uses a *postmortem* approach and employs the definition of a known or observed emergent property and aims to identify its cause, in terms of the states of system components and their interactions [Chen et al. 2009a; Seth 2008]. A key issue with this perspective is that a prior observation of an emergent property is required, and that emergent properties need to be defined in such a way that the macro-level can be reduced or traced back to the micro-level.

Current approaches [Chen et al. 2009a; Kubik 2003; Seth 2008] also have limiting assumptions and constraints when applied to larger systems in terms of the number and complexity of agents and their interactions. For example, most approaches do not consider mobile agents [Kubik 2003], assume unfeasible a priori specifications and definitions of emergent properties [Szabo and Teo 2012a], or do not scale beyond models with a small number of agents [Teo et al. 2013]. Finally, most studies focus only on postmortem observation of emergence when a tangible representation of the system is available for examination [Gore and Reynolds 2008; Moncion et al. 2010], and there is little work in predicting that emergent behavior will happen or identifying it as it happens. In the multiagent systems community, approaches focus on the engineering of systems to exhibit emergent behavior [Bernon et al. 2003; Jacyno et al. 2009; Salazar et al. 2011], without a subsequent analysis of potential side effects of the engineered emergent property.

Kubik [2003] proposed a method for formalizing weak emergence without prior knowledge of emergence using an abstraction of agents as Chomsky grammars [Chomsky 1956]. The approach adopts a set-based view on the world, where emergence is defined as the difference between a calculated and an observed behavior. Specifically, emergent behavior is defined as those system states that cannot be obtained from the summation, using a superimposition operator, of individual agents' behaviors. This is a broad definition that does not consider the actual behavior of the system under study. Moreover, the calculation of the expected behavior suffers from state explosion and renders the method infeasible for practical use [Szabo and Teo 2012b]. In this article, we expand on this approach by proposing a grammar-based formalism and process with a new definition of emergence that distinguishes possible system states and emergent properties of interest from the larger set of all combinations of system states. In contrast to Kubik's approach, our formalism allows for different agent types, includes mobile agents, and covers open systems where the number of agents varies over time. The main contributions of our work include:

—A formal approach for identifying weak emergence as it is happening in a multiagent system: our approach relies on the abstractions of complex agents as formal grammars that produce outputs as words written on a common tape, and on the definition of the set of emergent property states as the set difference between a calculated system state and the current system state.
—The introduction of a degree of interaction measure to separate between the states in the emergent property set and to reduce state-space explosion as models grow in size, in terms of the number of agents, and complexity, in terms of the number of agent attributes and agent interactions.
—An extensive theoretical and experimental analysis of the feasibility of our approach and its application to large-scale multiagent systems.

The remainder of the article is organized as follows. We present our approach in Section 3. Section 2 reviews different perspectives, classifications, and formal studies of emergence. Section 4 illustrates our approach using an established model and presents an experimental analysis of our work, discussing its limitations. Section 5 concludes this article and discusses future work.

2. RELATED WORK

Emergence can be considered from two main perspectives, namely, philosophical and scientific. From a *philosophical* perspective, emergent properties are subjectively a product of the unexpected behavior of complex systems, the limitations of the observer's knowledge [Johnson 2006], the tools employed, and the scale and level of abstraction under which the system is observed [Bonabeau and Dessalles 1997]. *Scientific* *perspectives* [Abbott 2006; Kubik 2003] criticize the idea of temporary lack of knowledge of the observer and define emergent properties as intrinsic to a system and independent from the eye of the beholder [Crutchfield 1999]. Natural and social sciences explain emergence via theories in physics, biology, and human behavior among others, focusing on self-organization and hierarchy. Emergence is thus defined as the formation of order from disorder based on *self-organization* [Fromm 2007]; a property is emergent if it is discontinuous from the properties of the components at the lower levels in the *hierarchy* [Baas and Emmeche 1997]. Using modeling and simulation, we aim to predict, identify, classify, and reason about emergence. Prediction is done before the observation of emergence, while the identification of emergence highlights its presence. The classification of emergent behavior allows us to determine whether an emergent property is harmful or beneficial. Finally, reasoning enables the understanding of the cause and effect of emergence.

A macro-level property that characterizes the system as a whole is defined to be *weakly* emergent if it can be derived from the micro-level dynamics but only by a finitely long simulation [Bedau 1997]. Other definitions of weak emergence consider that the whole is constrained or influenced by the parts (*upward causation*), but that at the same time the parts are influenced by the whole (*downward causation*) [Bedau 2003; O'Conner 1994]. For example, when the general price level of goods and services rises, the cost of living increases. Higher living cost demands higher income, which in turn results in higher prices of goods and services. The cycle continues until a policy to decrease the inflation rate is introduced.

The undesired and unpredictable effects of emergent properties demand a formal and practical approach to understanding and identifying emergence. Simulation is considered to be a potential solution for the formal study of emergence [Bedau 1997; Darley 1994; Hovda 2008], with the advantage that it permits the development of methods that can be implemented in practice. For example, a practical method to check the upper bound of the R pentomino, which is a five-cell pattern in the Game of Life, is through simulation: after 1,103 time steps, we see that the R pentomino settles down to a stable state that fits into a 51-by-109 cell region [Bedau 2003]. According to Darley [1994], simulation is regarded as the most efficient way to predict emergent properties; Hovda [2008] quantifies emergence in terms of the amount of simulation needed to derive a fact. Agent-based modeling (ABM) is considered an appealing approach to model and simulate complex systems exhibiting emergence [Holland 1997]. ABM provides a detailed description of the system, including its components and their interactions, thus facilitating the detecting and reasoning of the cause and effect of emergence. Moreover, ABM is relevant to complex systems as both rely on the interaction of autonomous individual objects. In addition, several formalisms have been proposed to obtain or engineer emergent behavior, such as the DEVS extension proposed by Mittal [2013], but they have yet to be employed in practice.

Approaches for the identification and reasoning about emergent behavior can be classified into three main categories, namely, variable based, event based, and grammar based. Each has advantages and disadvantages, as we discuss in the following. In *variable-based* methods, one variable is chosen to model the attribute space that describes the state of the observed system. This variable is then used to detect and measure emergent properties [Norros et al. 2006]. Usually, emergence is measured using probability and information theory [Fisch et al. 2010; Mnif and Muller-Schloer 2006; Seth 2008]. For example, the change of the center of mass of a group of birds may indicate the formation of flocking behavior.

Many variable-based efforts [Gabbai et al. 2005; Holzer et al. 2008; Mnif and Muller-Schloer 2006; Wolf et al. 2005] employ Shannon entropy [Shannon 2000], which measures the uncertainty and unpredictability of a system with respect to one attribute.

The key idea is that emergence most likely occurs as the system self-organizes and exhibits some kind of pattern or structure, thus resulting in lower entropy. Mnif and Muller-Schloer [2006] introduce emergence as the difference between the entropy at the beginning and at the end of the system run. A system is said to exhibit emergence if the entropy difference is positive; that is, the entropy value decreases in the end. Despite its simplicity, Shannon entropy only considers a single system attribute with discrete values. To address systems containing many attributes with continuous values, Fisch et al. [2010] define multivariate divergence as "an unexpected or unpredictable change of the distribution underlying the observed samples" using Hellinger distance as a measure of emergence. This measurement suffers from expensive computation of density functions and requires significant intervention from the user. Inspired by the idea that weak emergence is both dependent upon and autonomous from the micro-level causal factors, Seth [2008] proposes G-emergence as a measure of emergence based on two other nonlinear time series quantities, G-causality and G-autonomy, which compute the dependence and autonomy of a variable with respect to a set of other variables, respectively. A macro-variable M is *G-emergent* from a set of micro-variables m if and only if M is G-caused and G-autonomous with respect to m. However, a set of micro-variables must be defined, and the computations of G-causality, G-autonomy, and G-emergence are expensive. One of the most significant drawbacks of variable-based emergence formalization is that it requires prior knowledge of emergence to define one or more variables that capture the system behavior.

In *event-based* approaches [Chen et al. 2009b], emergence is defined as complex events that can be reduced to a sequence of simple events. An event is a state transition occurring at a particular level of abstraction. A *simple* event results from the execution of a single state transition rule. A *complex* event is either a simple event or two complex events satisfying a set of constraints with respect to each other. Similar to variable-based approaches, event-based approaches need both knowledge about the emergent behavior and a specification of the formalism of event types to be defined in advance, and thus can be applied only for the postmortem analysis of emergence.

To overcome the challenging problem of the need for a prior identification of emergence, Kubik [2003] proposes a *grammar-based* approach. An extended cooperating array grammar system is used to formalize the environment, agents, and their cooperation. The system behavior is the language, that is, the set of words, resulting from derivations of grammars (agents) on the tape (two-dimensional array of symbols). The approach is based on the idea of emergence as defined by "the whole is more than the sum of its parts." An emergent property is defined as a system state that results from the interactions of agents and that cannot be produced by summing the individual agent states. Given an initial system state, two languages L_{WHOLE} and L_{PARTS} are calculated. The grammar systems generate the languages that can in turn be represented as sets of "words" to encode agent and system attributes. L_{WHOLE} defines a set of system states produced by the system agents acting together and interacting as a whole. L_{PARTS} presents the sum of the agents' behaviors without considering the interactions among them. As a result, emergence is defined as $L_{WHOLE} - L_{PARTS}$. The approach is exemplified using a small Game of Life example. The grammar-based approach does not require a priori definition of emergence and presents an elegant formalism that could be easily automated, although this is not considered by the initial article. However, four main limitations exist: (1) there is no explicit provision for agent type; (2) it is unclear how to deal with mobile agents; (3) in the Game of Life example, agents are stationary, and the number of agents is always equal to the number of cells, which is an unrealistic assumption; and (4) the state spaces generated by calculating L_{WHOLE} and L_{PARTS} are unnecessarily large. In our work, we address these limitations as discussed later.

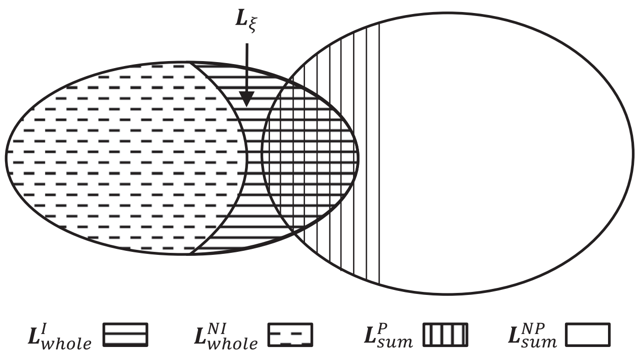

Fig. 1. Set of emergent property states.

3. PROPOSED APPROACH

This section presents our proposed approach for the identification of emergence. We advance a formalism for the definition of multiagent systems that allows for the identification of emergence as a set difference between the set of system states obtained from the interaction of all agents in the system and a set of states calculated from the aggregated individual agent behaviors. We present relevant background and give an overview of our approach, and then we define our formalism in detail. We present a theoretical analysis of our approach, showing how it also suffers from state-space explosion, similar to existing methods. We then introduce a construct that significantly reduces the state space and allows us to analyze much larger systems.

3.1. Background and Overview

Our approach for the identification of emergent properties follows Kubik's grammar-based approach [Kubik 2003] to calculate L_ξ, the set of emergent property states, as the difference:

$$L_\xi = L_{whole} - L_{sum}, \tag{1}$$

where L_{whole} describes all possible system states due to agent-to-agent and agent–environment interaction, and L_{sum} is the sum of all individual agent behaviors, without considering agent interactions. This broad perspective of emergence, as in Kubik's approach, leads to state explosion when determining L_{sum} and L_{whole}. This is because all possible combinations of individual agent states are considered following a defined superimposition operator, without including system-defined rules.

We propose a new perspective that significantly reduces the state space for L_ξ. First, it is important to highlight here that while Kubik refers to the difference $L_{whole} - L_{sum}$ as emergence, we refer to this set as the *emergent states set*, because it is from this set that emergent properties can be extracted. This captures the insight found in the literature that emergent behavior has a temporal aspect [Gore and Reynolds 2008]. Thus, an emergent property can be inferred or deduced from the set of emergent property states. For example, a flocking emergent property in a flock-of-birds model can be characterized by a number of states that show birds close to each other, with various degrees of clustering.

Second, we observe that the size of L_{whole} is dependent on the number of interactions and state transition rules defined by a modeler and represents a subset of interest (to the modeler) from all the rules in a given system. This would be the case, for example, when we consider different kinds of rules in modeling a flock of birds: the entire rule set, or some rules of interest while ignoring others. As shown in Figure 1, L_{whole} can be redefined as

$$L_{whole} = L^{I}_{whole} \cup L^{NI}_{whole}, \tag{2}$$

where L^{I}_{whole} is bounded by the number of interaction and behavior rules that are of interest to the user for the particular study, and L^{NI}_{whole} represents the set of all possible system states that are not of interest. This definition captures the idea that a simulation model is an abstraction of the real system that is constructed by the modeler according to the purpose of the simulation study and implies either that the abstraction might ignore particular system aspects or that some behavior and interaction rules might be irrelevant for the study. These are design and modeling decisions that the modeler has to take.

The size of L_{sum} increases exponentially with the increase in the number of agents as all possible combinations of agent states are considered. Similarly, L_{sum} can be also redefined as follows:

$$L_{sum} = L^{P}_{sum} \cup L^{NP}_{sum}. \tag{3}$$

Although it is mathematically possible to compute the entire L_{sum} as the possible combinations of agent states, in practice, there is a subset of feasible agent behaviors, L^{P}_{sum}, and the remaining can be defined as L^{NP}_{sum}. Determining L^{P}_{sum} is not always straightforward because it requires analyzing whether a system state is possible according to some criteria. A potential approach to derive L^{P}_{sum} is to add constraints among agents and use these constraints to determine whether a system state is in L^{P}_{sum} or in L^{NP}_{sum}. For example, on a one-way street, if one agent abstracting a car is behind another car agent, in the next simulation iteration the order cannot be reversed. We showed in Section 3.4 that L^{P}_{sum} is not needed.

Given this, we redefine the set of emergent property states L_{ξ} as

$$L_{\xi} = L^{I}_{whole} - L^{P}_{sum}. \tag{4}$$

Emergent property states, with respect to a model of interest, are the result of nontrivial interactions among agents. These interactions lead to states that cannot be derived from summing individual agents' states. There are states in L^{I}_{whole} that can be found in L^{P}_{sum} and thus

$$L^{I}_{whole} \cap L^{P}_{sum} \neq \emptyset. \tag{5}$$

Intuitively, these states are resultant from agent computation that does not require interactions or from the interactions of agents that have no effect on agent behavior or their overall effect cancels out. For example, in the flock-of-birds model detailed in Section 4, two individuals can be very far apart and as such interaction rules between them have no effect.

Contrary to Kubik's approach, which regards emergent properties as all system states in L_{ξ}, we do not consider that L_{ξ} contains the emergent behavior but only states that together, following particular criteria, can form an emergent property. As discussed earlier, this follows the intuition that the emergent property may take a while to manifest and thus may be composed of more than one emergent state, following a particular criterion, for example, flocking. In the next step, we propose to determine whether L_{ξ} contains emergent properties that have been seen before or that are beneficial or harmful to the system. We discuss in Section 4.4 an example of how this can be achieved. In this article, we show the process of deriving the set of emergent property states L_{ξ}.

In addition to a more specialized definition of emergence, we enhance the grammar-based approach with three main extensions: (1) introduce agent type (A_{ij} denotes an

Table I. Glossary of Notations

	Notation	Description
System	$S(t)$	State of system at time t
Environment	V_E	Set of possible cell states
	V_e	Set of possible states of cell e
	$S_E(t)$	State of environment at time t
	$s_e(t)$	State of cell e at time t
Agent Type	m	Number of agent types
	n_i	Number of agents of type i $(1 \leq i \leq m)$
	V_{A_i}	Set of possible states for agents of type i
Agent	n	Number of agents
	A_{ij}	Agent of type i $(1 \leq i \leq m)$ and instance j $(1 \leq j \leq n_i)$
	V_A	Set of possible agent states for all agent types
	P_i	Set of attributes for agents of type i
	R_i	Set of behavior rules for agents of type i
	$s_{ij}(t)$	State of agent A_{ij} at time t
	$L(A_{ij})$	Set of system states representing the behavior of agent A_{ij}
Emergence	$\oplus$	Superimpose operator
	L_{whole}	Set of system states representing the behavior of the system as a whole
	L_{sum}	Set of system states representing the sum of the agents' behaviors
	L_{ξ}	Set of emergent property states

agent instance j of type i); (2) introduce mobile agents by defining mobility as agent attributes $P_i = P_{i\_mobile} \cup P_{i\_others}$, where P_i denotes the set of attributes of the agent instance i; and (3) agents enter and leave the system to model open systems. As a result, the proposed approach has a wide variety of applications in which system components belong to different types, can move in space, and have multiple attributes. These kinds of systems are ubiquitous in practice and include traffic networks [Manley and Cheng 2010] and social networks [Haglich et al. 2010]. Our proposed approach consists of two major steps, namely, modeling the system using our proposed formalism, followed by the application of our method for emergence identification as the system is running. To increase the usability of our formalism, we would like to further this work by having other agent formalisms translated into our proposed formalism. This effort is part of our future work. We present our formalism in the following.

3.2. Emergence Formalism

A multiagent system consists of m different types of n agents (A) interacting in an environment (E). A list of notations is shown in Table I.

The environment (E) is a part of the system that lies outside the agents and can be regarded as a platform for agent interactions. For simplicity, we assume that E has no behavior rules and that it changes only as a result of agent actions. Changes of the environment, in turn, impact the agent behavior. E is modeled as a two-dimensional (2D) grid[1] that is subdivided into c units called cells, denoted by e. Changes in the environment are therefore changes of the states of cells. For example, a cell turns from "occupied" to "free" when the agent that occupies the cell moves to another cell. V_e denotes the set of possible states of cell e. Similarly, V_E denotes the set of possible cell states, and $V_e \subseteq V_E$. In addition, the environment state is made up of the states of all cells. $s_e(t)$ and $S_E(t)$ denote the state of cell e and the environment E at time t, respectively.

[1]E can be easily extended to model other topologies.

Agents are autonomous entities characterized by a set of attributes. The values of these attributes at a point in time represent the state of an agent at that time. The state of an agent changes as dictated by the behavior of the agent. Agents act and interact with other agents and the environment according to agent-specific rule sets, which contain behavior rules defined as follows:

$$rule(condition) : s_{ij}(t) \rightarrow s_{ij}(t+1) \tag{6}$$

for agent A_{ij}, $s_{ij}(t) \in S(t), \forall t \geq 0$. If the *condition* is fulfilled at time t, the agent will apply the rule to transform its current state at time t, $s(t)$, to a state at the next time step $s(t+1)$. Rule conditions may include the state of the agent in terms of attribute values, but also the states of other agents, as well as logical expressions containing both agent- and system-specific functions. The neighbors of an agent are close to the agent in terms of some proximity. The definition of this proximity is problem specific; different models of the same problem may also specify the neighborhood in different ways. For example, cellular automata have two main types of neighborhoods: the *von Neumann* neighborhood and the *Moore* neighborhood. The former consists of the four orthogonally adjacent cells. The latter includes eight neighbors inhabiting the cells that are horizontally, vertically, and diagonally adjacent to the cell whose state is to be calculated. We define a neighboring function *neighbor*(a, b), which returns true if agents a and b are neighbors. In addition, if *condition* contains a *neighbor* function, then the behavior rule is called a *neighboring behavior rule*, and an *individual behavior rule* otherwise.

We distinguish between different types of agents, and, for an agent of type i, we note P_i as the set of attributes for agent i, with P_{i_mobile} as the attributes that model mobility. In summary, the agent behavior is characterized by behavior rules that define how agents interact with other agents and the environment. We assume that no evolutionary processes are involved in the system; that is, behavior rules do not change over time. R_i denotes the set of behavior rules of agents of type i. R_i consists of R_{i_mobile} that impacts agent mobility, that is, changes values of attributes of P_{i_mobile}, and R_{i_others} for the rest. An agent changes its state as a condition is met. The state of A_{ij} at time t, denoted by $s_{ij}(t)$, is defined by values of its attributes at time t. V_{A_i} denotes the set of possible agent states for agents of type i.

3.3. System Formalism

A multiagent system consisting of a set of agents and their environment is formalized as an extended cooperating array grammar system where Chomsky grammars [Chomsky 1956] represent agents, and a two-dimensional array of symbols represents the global environment. A Chomsky grammar is formally defined as

$$G = (N, T, P, S),$$

where N is a nonterminal alphabet, T is a terminal alphabet, P is a set of rewriting rules (productions) in the form $x \rightarrow y$, and $S \in N$ is a starting symbol. The language generated by G is a construct

$$L(G) = \{w | S \Rightarrow Gw, w \in T^*\}.$$

The elements of the formal language are called strings or words. A cooperating array grammar system is a construct of grammars (agents) mutually rewriting strings of symbols on a common tape [Hopcroft et al. 2001]. Each agent rewrites a portion of the tape as is usual in the case of individual agents. Grammars work with different cooperation strategies that describe how they are allowed to rewrite the symbols on the tape and whether the agents can communicate directly. The abstraction of an agent as a grammar system follows from the idea that the agent output to all other agents can be interpreted as a communication language. This communication language is generated by a grammar, which can thus abstract an agent [Kubik 2003]. An agent's language is composed of the set of words that the agent outputs on the tape. Each agent behavior or grammar has its own rewriting rules defining how the grammar cooperates with the other grammars and with the 2D environment representing the array.

A system of m agent types and a total of n agents $A_{11}, \ldots, A_{1n_1}, \ldots, A_{mn_m}$ interacting in a 2D grid environment of c cells is defined as follows:

$$GBS = (V_A, V_E, A_{11}, \ldots, A_{1n_1}, \ldots, A_{mn_m}, S(0)), \tag{7}$$

where V_A denotes the set of possible agent states for all agent types, V_E denotes the set of possible cell states, A_{ij} denotes an agent of type i ($1 \leq i \leq m$) and instance j ($1 \leq j \leq n_i$), and $S(0)$ denotes the initial system state.

For the environment (E),

$$V_E = \bigcup_{e=1}^{c} V_e, \tag{8}$$

where V_e denotes the set of possible states of cell e. The state of the entire environment is made up of the states of all its cells. The states of cell e and the environment E at time t are $s_e(t) \in V_e$ and $s_E(t) \in V_E$, respectively.

For the agents (A),

$$V_A = \bigcup_{i=1}^{m} V_{A_i}, \tag{9}$$

where V_{A_i} denotes the set of possible states for agents of type i, $V_{A_i} \in V_A$.

Agent of type i ($1 \leq i \leq m$) and instance j ($1 \leq j \leq n_i$), A_{ij}, is defined as follows:

$$A_{ij} = (P_i, R_i, s_{ij}(0)), \tag{10}$$

where P_i denotes the set of attributes for agents of type i, R_i denotes the set of behavior rules for agents of type i, and $s_{ij}(0)$ denotes the initial state of the agent. P_i is defined as

$P_i = P_{i\_mobile} \bigcup P_{i\_others}$, where

$P_{i\_mobile} = \{\text{x} \mid \text{x is an attribute that models mobility}\}$

$P_{i\_others} = P_i \setminus P_{i\_mobile}$

Agents change states according to behavior rules R_i that are defined as a set of functions. The rules that affect the mobility of agents, for example, a change in location or speed, are defined as mobile rules, R_{i_mobile}.

$R_i = \{rule | rule : \{True, False\} \times V_{A_i} \rightarrow V_{A_i}\}$

$condition : V_A \rightarrow \{True, False\}$

$R_i = R_{i\_mobile} \bigcup R_{i\_others}$

The behavior rules set is defined as a set of rules formed by a condition function and an agent state, capturing that the rule is applied if the agent is in a particular state in V_{A_i} and the condition function is *True*. The condition function is defined over the system states, thus capturing other agent states (including the agent neighborhood).

A_{ij} has an initial state $s_{ij}(0) \in V_{A_i}$. The system state at time t ($S(t)$) is composed of the state of the environment ($s_E(t)$) and states of agents ($s_{ij}(t)$) at time t. Hence,

$$S(t) = s_E(t) \bigcup_{\forall i \forall j} s_{ij}(t). \tag{11}$$

3.4. Proposed Process for Emergence Identification

Since emergent properties are becoming de facto as systems increase in complexity, we believe that the study of emergence should be factored in as part of any modeling and simulation study as a coherent approach to understand the system. In this process, we assume that our proposed formalism is used from scratch to model a system. We discuss in Section 4.4 potential solutions for the case whereby other formalisms are employed. Our approach for the identification of emergence computes two sets of system states corresponding to the two levels of abstraction defined earlier, namely, the macro-level when regarding the system as a whole (L_{whole}) and the micro-level when regarding the system as an aggregation of its individual agents (L_{sum}). Emergent system states are defined as the difference between L^I_{whole} and L^P_{sum}:

$$L_\xi = L^I_{whole} - L^P_{sum}.$$

In the following, we focus on the calculation of L^I_{whole} and L_{sum} and leave the approach to obtain L^P_{sum} as future work.

Taking the interactions of agents (denoted as *GROUP*) into account, the system behavior of interest (L^I_{whole}) returns a set of words (w) that represents the set of system states reachable from the initial system state. Given an initial state, the system is simulated until it arrives in a state that has already appeared before. L^I_{whole} with respect to the initial state is therefore a set of all distinct states obtained as follows:

$$L^I_{whole} = \{w \in V^{c+n} \mid S(0) \Rightarrow^*_{GROUP} w\}. \tag{12}$$

While L^I_{whole} captures the behavior of the system as the agents are interacting, L_{sum} represents an aggregation of the individual agent behaviors when no interaction between agents is considered, as if the agents were executing alone in the environment. Informally, a superimposition operation $\oplus$ to capture the aggregated words on the common tape is introduced for the calculation of L_{sum}:

$$L_{sum} = \oplus(L(A_{11}), \ldots, L(A_{1n_1}), \ldots, L(A_{mn_m})), \tag{13}$$

where $L(A_{ij})$ denotes the behavior of agent A_{ij}. To obtain these aggregated words that are part of L_{sum}, we employ the superimposition operator [Kubik 2003] as outlined later.

As an example, let $w_1 = a_1a_2\ldots a_x, w_2 = b_1b_2\ldots b_y$ be words of symbols over an alphabet $V = V_A \cup V_E$, and ϵ denotes the empty symbol. Hence, the superimposition of the word w_1 on the word w_2 is a function $\oplus : V^* \times V^* \times V^* \ldots \times V^* \to V^*$ that results in $w_{supimp} = c_1c_2\ldots c_z$ defined as follows:

1. $z = max(x, y), 1 \leq k \leq z$, $\times$ is the Kleene star operator;
2. if $a_i \in V_A$, then $c_k = a_i$;
3. if $a_i = \epsilon$, then $c_k = b_j$;
4. if $b_j = \epsilon$, then $c_k = a_i$;
6. if $a_i \in V_E$ and $b_i \in V_E$, then $c_k = a_i$;
7. if $a_i \in V_E$ and $b_j \in V_A$, then $c_k = b_j$.

The superimposition occurs when two agents, when considered individually, write two words (w_1 and w_2 from earlier) on the same position of the tape. The superimposition is done over all permutations of n behaviors of agents. It is important to note here that ordering is important in the process of calculating the superimposition. For instance,

$$\begin{aligned}\oplus(L_1, L_2, L_3) = L_1 \oplus (L_2 \oplus (L_3)) \cup L_1 \oplus (L_3 \oplus (L_2))\\ \cup L_2 \oplus (L_1 \oplus (L_3)) \cup L_2 \oplus (L_3 \oplus (L_1))\\ \cup L_3 \oplus (L_1 \oplus (L_2)) \cup L_3 \oplus (L_2 \oplus (L_1)).\end{aligned}$$

Defining the sum of the individual agents' behaviors is difficult. Ideally, the result should contain exactly all designed system states that can be derived from the system specification. Unfortunately, this is only true if agents are independent from the others in the system. Given the initial system state, we obtain n states where each consists of one agent. Considering only one agent in the system, the agent's behavior returns a set of words (w) that represents the set of system states reachable from the corresponding system state. $L(A_{ij})$ is defined as follows:

$$L(A_{ij}) = \{w \in V^{c+1} \mid (s_E(0) \cup s_{ij}(0)) \Rightarrow^* w\}. \tag{14}$$

Agent symbols have priority over environmental symbols. Any nonempty symbol has priority over the empty symbol ϵ. For example, consider $V_A = \{a_1, a_2, a_3\}$, $V_E = \{o, f\}$, and $L_1 = \{fa_1of\}, L_2 = \{oa_2ff\}, L_3 = \{foffa_3\}$ languages for three agents. Then, assuming that all symbols are generated when the agents are executing in isolation, the superimposition of the agents' behaviors will be the language $L_{sum} = \{fa_1ofa_3, oa_2ffa_3, fa_1ffa_3, fa_2ffa_3\}$. The sum of agents' behaviors (L_{sum}) is defined as the set of words resulting from superimposing behaviors of individual agents:

$$L_{sum} = \oplus(L(A_{11}), \dots, L(A_{1n_1}), \dots, L(A_{mn_m})), \tag{15}$$

where $L(A_{ij})$ denotes the behavior of agent A_{ij} and $\oplus$ is the superimpose operator.

3.4.1. Implementation. Taking the interactions of agents (denoted as *GROUP*) into account, the system behavior of interest (L^I_{whole}) returns a set of words (w) that represents the set of system states reachable from the initial system state. Algorithm 1 presents the pseudo-code for the calculation of L^I_{whole}. Given an initial state $S(0)$ (line 2), the system is simulated until it reaches a state that has already appeared before (line 6). We stop the simulation when the system state repeats itself. L^I_{whole} comprises all distinct states $S(t)$ obtained (line 17). L_{sum} is calculated following the superimposition operation definition from Equation (15). We show an example of L_{sum} calculation in Section 4.

3.4.2. Theoretical Analysis. The complexity of deriving L_ξ ($O(L_\xi)$) consists of two parts: complexity of L^I_{whole} and complexity of L_{sum}:

$$O(L_\xi) = O\big(L^I_{whole}\big) + O(L_{sum}). \tag{16}$$

Because detecting emergence is to differentiate system states that appear when taking into account interactions of agents but not when regarding them separately, it is reasonable to use the number of system states as a complexity measure. Key complexity factors include environment size (2D grid of size x by y), number of agent types (m), number of agents (n), and the number of possible states an agent can take (s). We derive $O(L^I_{whole})$ and $O(L_{sum})$ in the *worst case*. Let $n = n' + n''$, where n' is the number of mobile agents, and n'' is the number of stationary agents.

$O(L^I_{whole})$: Given a position, an agent can take one of s states. Moreover, stationary agents are fixed in n'' positions. There are $\binom{xy-n''}{n'}$ possibilities for allocating n' mobile agents into the remaining $xy - n''$ positions. Hence,

$$O\big(L^I_{whole}\big) = O\left(\binom{xy-n''}{n'}s^n\right). \tag{17}$$

ALGORITHM 1: Pseudo-Code for L^I_{whole} Calculation

```
1: t:= 0;                                          //initialized clock
2: set S(0);                                       //set initial system state
3: $L^I_{whole}$:= Ø;
4: add S(0) to $L^I_{whole}$;
5: repeat:= false;
6: **while** repeat false **do**
7:     t:= t + 1;
8:     ...            //simulate next step
9:     //use a for loop to compare states
10:    **for** $i = 0$ to $t - 1$ **do**
11:        **if** $S(t)$ equal $S(i)$ **then**
12:            repeat = true;
13:            exit for loop;
14:        **end if**
15:    **end for**
16:    **if** repeat false **then**
17:        add $S(t)$ to $L^I_{whole}$ ;
18:    **end if**
19: **end while**
20: **return** $L^I_{whole}$;
```

$O(L_{sum})$: Without considering the interactions of agents, a mobile agent, in the worst case, can arbitrarily move to any position (cell) in the environment. Hence, the upper-bound complexity of superimposing individual behaviors for all agents is

$$O(L_{sum}) = O((xy)^{n'} s^n). \tag{18}$$

For example, in the Game of Life, agents are stationary, that is, $n'' = n$ and $n' = 0$, and $O(L^I_{whole}) = O(L_{sum}) = O(s^n)$). If all agents are mobile, that is, $n'' = 0$ and $n' = n$, then $O(L_{sum}) = O((xy)^n s^n)$ is much larger than $O(L^I_{whole}) = O(\binom{xy}{n} s^n)$. This is because the summing operation superimposes all combinations of individual agents' behaviors, including those that could never happen in practice, as discussed earlier.

3.5. State-Space Reduction

As shown previously, the state space grows exponentially when calculated as the set difference $L_{whole} - L_{sum}$. The main cause of this growth is the calculation of L_{sum}, which requires all possible combinations of system states to be computed. A key observation is that the definition of $L_\xi = L_{whole} - L_{sum}$ (and our proposed extension) implies that emergent property states in L_ξ are those states that are in L_{whole} but not in L_{sum}. Thus, if states can be identified with confidence as being in L_{whole} but not in L_{sum} *without* computing L_{sum}, the complexity of the approach will be greatly reduced, and the computation cost will become acceptable.

We propose to use the degree of interaction between agents, δ, as a useful criterion, thus eliminating the unnecessary calculation of L_{sum}. We define a property as emergent if the agent interaction producing that property is strong. In other words, compared to the situation where agents behave independently, agents with interactions impose a remarkable effect on the system state. Intuitively, more interactions lead to more changes of the state of the system, and it is largely accepted that interaction is a key prerequisite of emergence [Chan 2010; Holland 1997; Kubik 2003]. Moreover, emergence happens.

Thus, the system states of interest can be divided into L_o, due to interactions that are not strong, and L_ξ, due to strong interactions. There are three possible categories of nonstrong interactions, namely, no interaction (individual agent behavior), interaction that has no overall effect on the system, and weak interaction. Interaction can also be classified as direct and indirect [Birdsey and Szabo 2014], but regardless of the classification, the interactions between agents trigger state changes [Melo and Veloso 2009; Jacyno et al. 2009]. For simplicity, we propose to use system state change as a measure of the degree of agent interaction, following the intuition that strong interactions tend to effect significant changes on the state of the system.

3.5.1. Degree of Interaction Formalism. As defined earlier, we distinguish the behavior rules R_i into neighboring and individual, based on whether they consider the neighbors of an agent or are focused on individual agents, respectively. Interaction arises from *neighboring behavior rules* that define how an agent interacts with other agents. For example, a bird changes its position because of coordination (collision avoidance, alignment, and cohesion) with other neighboring birds. Besides neighboring rules, an agent also has *individual rules* that govern the agent behavior when the agent acts individually without interacting with its neighbors. For example, a bird may not change its speed for a period during which it is alone in the environment. As a result, the difference between two system states $S(t)$ and $S(0)$ at time t and zero, respectively, $D(S(t), S(0))$, is due to individual rules, $D_I(S(t), S(0))$, and neighboring rules, $D_N(S(t), S(0))$:

$$D(S(t), S(0)) = D_I(S(t), S(0)) + D_N(S(t), S(0)). \tag{19}$$

$D_N(S(t), S(0))$ represents the degree of interactions between agents. Because emergent properties arise from interaction, and agent interaction, in turn, arises from neighboring rules, $D_N(S(t), S(0))$ can be regarded as a measure or a criterion for determining emergence. Consider system state $S(t)$ in L^I_{whole}:

$$S(t) \in \begin{cases} L_o & \text{if } D_N(S(t), S(0)) = 0, \text{ no interaction or no effect} \\ & \text{if } 0 < D_N(S(t), S(0)) \leq \delta, \text{ weak interaction} \\ L_\xi & \text{otherwise,} \end{cases}$$

where δ $(0 < \delta < 1)$ is a predefined threshold. Algorithm 2 presents the pseudo-code for deriving L_ξ based on the degree of agent interaction. L_o and L_ξ are initialized to empty sets (line 2 and line 3, respectively). The difference between every state in L^I_{whole} and the initial state is measured (line 5) and compared with the predetermined threshold δ. If the difference is less than or equal to δ, the corresponding state is added to L_o. Otherwise, it is added to L_ξ, that is, an emergent property state. The difference between $S(t)$ and $S(0)$ is accumulated from the difference between $s_{ij}(t)$ and $s_{ij}(0)$ for individual agents (line 9). The *for* loop terminates as soon as the accumulated difference is larger than δ (line 12).

3.5.2. $D_N(S(t), S(0))$ Calculation. The system state is composed of the states of all constituent agents and the states of the environment. Therefore, we define

$$D_I(S(t), S(0)) = \frac{1}{n} \sum_{\forall i \forall j} d_I(s_{ij}(t), s_{ij}(0)) \tag{20}$$

$$D_N(S(t), S(0)) = \frac{1}{n} \sum_{\forall i \forall j} d_N(s_{ij}(t), s_{ij}(0)) + D_N(S_E(t), S_E(0)), \tag{21}$$

where n denotes the number of agents; $d_I(s_{ij}(t), s_{ij}(0))$ and $d_N(s_{ij}(t), s_{ij}(0))$ denote the difference between A_{ij}'s states in $S(t)$ and $S(0)$ due to individual rules and neighboring rules, respectively; and $D_N(S_E(t), S_E(0))$ denotes the difference between states of the

ALGORITHM 2: Pseudo-Code for L_ξ Calculation

```
 1: procedure calculate_$L_\xi$
 2:     $L_o = \emptyset$;
 3:     $L_\xi = \emptyset$;
 4:     set $\delta$;                                   //set weak emergence threshold
 5:     for each state $S(t) \in L^I_{whole}$ do
 6:         $D(t,0) := 0$;
 7:         for each entity $A$ in system do
 8:             call $d_A(t,0)$;                        //entity state different time $t$ and zero
 9:             add $d_A(t,0)$ to $D(t,0)$;
10:             if $D(t,0) > \delta$ then               //$L_\xi$ state
11:                 add $S(t)$ to $L_\xi$;
12:                 exit for loop;
13:             end if
14:         end for
15:         if $D(t,0) = 0$ then
16:             add $S(t)$ to $L_o$;                    //no interaction
17:         else if $D(t,0) \leq \delta$ then
18:             add $S(t)$ to $L_o$;                    //weak interaction
19:         end if
20:     end for
21:     return $L_\xi$;
22: end procedure
```

environment at time t and zero. Let

$$d(s_{ij}(t), s_{ij}(0)) = d_I(s_{ij}(t), s_{ij}(0)) + d_N(s_{ij}(t), s_{ij}(0)) \tag{22}$$

be the difference between A_{ij}'s states in $S(t)$ and $S(0)$ in total, that is, due to both individual and neighboring rules.

From Equation (21), to calculate $D_N(S(t), S(0))$, we need $d_N(s_{ij}(t), s_{ij}(0))$, which requires $d(s_{ij}(t), s_{ij}(0))$ and $d_I(s_{ij}(t), s_{ij}(0))$ as from Equation (22). We propose a method to measure $d(s_{ij}(t), s_{ij}(0))$ and $d_I(s_{ij}(t), s_{ij}(0))$. The calculation of $D_N(S_E(t), S_E(0))$ will be discussed after that. Because the state of an agent is characterized by the values of its attributes, which may have different units of measurement and different ranges, we need to normalize agent attributes to the same range [0,1] with no units. For example, the speed of a car may vary from 0km/h to 100km/h, while its travel time is measured in seconds. Interval [0,1] of normalized attributes ensures that agent state difference $d_N(s_{ij}(t), s_{ij}(0))$ and system state difference $D_N(S(t), S(0))$ due to neighboring rules are also in interval [0,1]. An attribute p is normalized using the following scaling formula:

$$p' = \frac{p - p_{min}}{p_{max} - p_{min}}, \tag{23}$$

where p_{min} and p_{max} are the minimum and maximum values, respectively, of the attribute p. In addition, we assume that all agent attributes can be measured in a numerical form. Nonnumerical attributes could be translated into a numerical form. For example, a color attribute could have its Red Green Blue (RGB) values translated into numbers. This translation is problem specific, and we leave this issue for future work. Thus, $d(s_{ij}(t), s_{ij}(0)$ is defined as

$$d(s_{ij}(t), s_{ij}(0)) = \frac{\sum_{p \in P_i} |p'(t) - p'(0)|}{|P_i|}, \tag{24}$$

where P_i denotes the set of attributes for agents of type i, and $p'(t)$ denotes value of normalized attribute p at time t. Algorithm 3 presents the pseudo-code for the calculation of $d(s_{ij}(t), s_{ij}(0))$. This difference counts the differences of the values of attributes at time t and zero. Agent attributes are normalized in line 4.

ALGORITHM 3: Pseudo-Code for $d(s_{ij}(t), s_{ij}(0))$ Calculation

```
1: procedure calculate_$d(s_{ij}(t), s_{ij}(0))$
2:     $d(s_{ij}(t), s_{ij}(0)) := 0$;
3:     for each attribute $p \in P_i$ do
4:         add $|(p(t) - p(0)|/(p_{max} - p_{min})$ to $d(s_{ij}(t), s_{ij}(0))$;
5:     end for
6:     $d(s_{ij}(t), s_{ij}(0)) := d(s_{ij}(t), s_{ij}(0))/|P_i|$;
7:     return $d(s_{ij}(t), s_{ij}(0))$;
8: end procedure
```

$d_I(s_{ij}(t), s_{ij}(0))$ is due to individual rules and defined as:

$$d_I(s_{ij}(t), s_{ij}(0)) = d(s, s_{ij}(0)), \tag{25}$$

where s is an agent state reached from $s_{ij}(0)$ after t steps considering only individual rules for A_{ij}; that is, A_{ij} is alone in the system:

$$s_{ij}(0) \xrightarrow{\text{individual rules}}{}^{t} s_{ij}(t) \tag{26}$$

If $s_{ij}(0) \xrightarrow{\text{individual rules}}{}^{t} s_{ij}(t)$, then $d(s_{ij}(t), s_{ij}(0)) = d_I(s_{ij}(t), s_{ij}(0))$ and $d_N(s_{ij}(t), s_{ij}(0)) = 0$.

For example, consider the flock-of-birds model discussed in more detail in Section 4. A bird has two key attributes: position, which is the location of the cell occupied by the bird, denoted by a 2D vector $(x(t), y(t))$, and velocity, denoted by a 2D vector $(\alpha(t), \beta(t))$. We assume that a bird does not change its velocity when flying alone and visits certain cells along its path corresponding to its velocity. Let $\sqrt{c}$ be the width/height of the environment, where c is the size, that is, number of cells, of the square environment 2D grid. $s_{ij}(0) \xrightarrow{\text{individual rules}}{}^{t} s_{ij}(t)$ if and only if:

1. $x(t) - x(0) = 0\ mod(\sqrt{c})$ and $y(t) - y(0) = 0\ mod(\sqrt{c})$;
2. $(\alpha(t), \beta(t)) = (\alpha(0), \beta(0))$,

where *mod* is the modulo operation.

3.5.3. $D_N(S_E(t), S_E(0))$ Calculation. The difference between states of the environment at time t and time zero is defined as follows:

$$D_N(S_E(t), S_E(0)) = min(D_{N,A_{ij}}(S_E(t), S_E(0))), \tag{27}$$

where $D_{N,A_{ij}}(S_E(t), S_E(0))$ denotes the difference of states of the environment at time t and time zero due to the behavior of agent A_{ij} only. Considering A_{ij} alone in the system, $D_{N,A_{ij}}(S_E(t), S_E(0))$ can be computed in a manner similar to Equation (24) with respect to all attributes of the environment.

4. FLOCK-OF-BIRDS MODEL

The boids model [Reynolds 1987] captures the motion of bird flocking and is a seminal example for studying emergence [Chan 2010]. We have applied our approach to models of traffic jams and the Game of Life, but because of its simplicity, we employ the flocks of birds model in this article. At the macro-level, a group of birds tends to move

Table II. Vector Representation for Velocity of Ducks

	Speed		
Direction	0	1	2
North	(0,0)	(0,1)	(0,2)
Northeast	(0,0)	(1,1)	(1,2), (2,1), (2,2)
East	(0,0)	(1,0)	(2,0)
Southeast	(0,0)	(1,-1)	(1,-2) (2,-1), (2,-2)
South	(0,0)	(0,-1)	(0,-2)
Southwest	(0,0)	(-1,-1)	(-1,-2), (-2,-1), (-2,-2)
West	(0,0)	(-1,0)	(-2,0)
Northwest	(0,0)	(-1,1)	(-1,2), (-2,1), (-2,2)

in a V-like formation, which has aerodynamic advantages, obstacle avoidance, and predator protection, regardless of the initial positions of the birds. At the micro-level, each bird obeys three simple rules: (1) separation: steer to avoid crowding neighbors, (2) alignment: steer toward average heading of neighbors, and (3) cohesion: steer toward average position of neighbors.

To demonstrate our approach, we extended the boids model to include two types of birds, ducks and geese. For ease of discussion, we model a multiagent system with 10 birds with equal number of ducks and geese interacting in an environment represented as a 2D grid of 8×8 cells. Each cell can be *occupied* by a bird or *free*, and two birds cannot be located at the same cell at the same time. Birds have two attributes: position and velocity. Position and velocity model birds' mobility. The *position* of a bird is the location of the cell occupied. The *velocity* is a vector that contains speed and moving direction. Ducks can fly zero, one, or two cells per time step in one of eight directions: north, northeast, east, southeast, south, southwest, west, and northwest. Similarly, geese can fly at the maximum speed of three cells per time step. The vector representation for velocity of ducks is shown in Table II. Birds behave according to three rules: (1) separation: avoid collision with nearby birds, (2) alignment: fly as fast as nearby birds of the same type, and (3) cohesion: stay close to nearby birds of the same type.

4.1. System Formalism

The boids model is formalized as

$$GBS_{boid} = (V_A, V_E, A_{11}, \ldots, A_{15}, A_{21}, \ldots, A_{25}, S(0)),$$

where A_{1j} denotes a duck instance j $(1 \le j \le 5)$, A_{2j} denotes a goose instance j $(1 \le j \le 5)$, $V_A = V_{A_1} \cup V_{A_2}$ denotes the set of possible states for the ducks (V_{A_1}) and the geese (V_{A_2}), and V_E denotes the set of possible cell states. $S(t) \in (V_A \cup V_E)^+$ denotes system state at time t, and $S(0)$ denotes the initial system state at time zero.

For cell e, $V_e = \{o, f\}$, where o means occupied and f means free, and $s_e(t) \in V_e$. For the entire environment E, $V_E = \bigcup_{e=1}^{64} V_e = \{o, f\}$, where $64 = 8\times8$ represents the number of cells, and V_E^k is the k modulo 8 cell on row $k/8 + 1$ of the grid.

A duck instance $A_{1j}(1 \le j \le 5)$ is defined as follows:

$$A_{1j} = (P_1, R_1, s_{1j}(0)),$$

where

$P_1 = P_{1\_mobile} \cup P_{1\_others}$

$P_{1\_mobile} = \{g_{1j}, v_{1j}\}$

$P_{1\_others} = \emptyset$

$V_{A_1} = \{(x, y)|1 \leq x \leq 8, 1 \leq y \leq 8\} \times \{(\alpha, \beta)| -2 \leq \alpha \leq 2, -2 \leq \beta \leq 2\}$

$R_1 = R_{1\_mobile} \cup R_{1\_others}, R_{1\_others} = \emptyset$

$s_{1j}(t) \in V_{A_1}$

R_{1_mobile} defines the update of the velocity v_{1j} and the position g_{1j} of duck A_{1j} over time, based on a condition function, as defined by the formalism in Section 3.3:

$R_{1\_mobile} = \{rule : \{True, False\} \times V_{A_1} \to V_{A_1}\}$.

$rule(condition(g_{1j}(t), g_{1k}(t)), v_{1j}(t), g_{1j}(t)) = (v_{1j}(t+1), g_{1j}(t+1))$, where $v_{1j}(t+1)$, $g_{1j}(t+1)$ are calculated as shown here:

$condition(g_{1j}(t), g_{1k}(t)) = neighbor(g_{1j}(t), g_{1k}(t)), \forall j \neq k$

$P_{1\_mobile} = \{g_{1j}, v_{1j}\}$

$P_{1\_others} = \emptyset$

$V_{A_1} = \{(x, y)|1 \leq x \leq 8, 1 \leq y \leq 8\} \times \{(\alpha, \beta)| -2 \leq \alpha \leq 2, -2 \leq \beta \leq 2\}$

$R_1 = R_{1\_mobile} \cup R_{1\_others}, R_{1\_others} = \emptyset$

$s_{1j}(t) \in V_{A_1}$

We limit the speed of ducks to two cells per time step so that they cannot fly arbitrarily fast. Consequently, absolute values of the horizontal component (α) and the vertical component (β) of the velocity vector are bounded to two cells. Note that $sign(\alpha)$ and $sign(\beta)$ return signs of α and β, that is, 1 for positive and -1 for negative, respectively. Both velocity and position of birds are represented as 2D vectors; the update is therefore simply vector additions:

$(\alpha, \beta) = v_{1j}(t) + separation(A_{1j}) + align(A_{1j}) + cohesion(A_{1j})$

$v_{1j}(t+1) = (sign(\alpha)min(|\alpha|, 2), sign(\beta)min(|\beta|, 2))$

$g_{1j}(t+1) = g_{1j}(t) + v_{1j}(t+1)$

Separation: If an agent of type duck a is a neighbor of another duck or goose b, then a flies away from b:

$$separation(a) = \sum_{\forall b \in A_{ij}, b \neq a, neighbor(a,b)} a.position - b.position.$$

Alignment: Duck a changes velocity by $\lambda\%$ toward the average velocity of its neighboring ducks:

$$align(a, b) = \left(\frac{\sum_{\forall b \in A_{ij}, b \neq a, neighbor(a,b)}^{k} b.velocity}{k} - a.velocity \right) \times \frac{1}{\lambda}.$$

Cohesion: Duck a moves by $\gamma\%$ toward the center of its neighboring ducks:

$$cohesion(a) = \left(\frac{\sum_{\forall b \in A_{ij}, b \neq a, neighbor(a,b)}^{k} b.position}{k} - a.position \right) \times \frac{1}{\gamma}.$$

$neighbor(a, b)$ determines if two agents are neighbors, in this case if $distance(a, b) \leq \epsilon$, that is, the birds are within ϵ cells of each other. $a.position$ and $a.velocity$ capture, for ease of reading, the position (g) and velocity (v) agent variables of agent a as defined earlier. The model for geese agents follows in a similar manner except that geese are blue, and their maximum speed is three cells per time step.

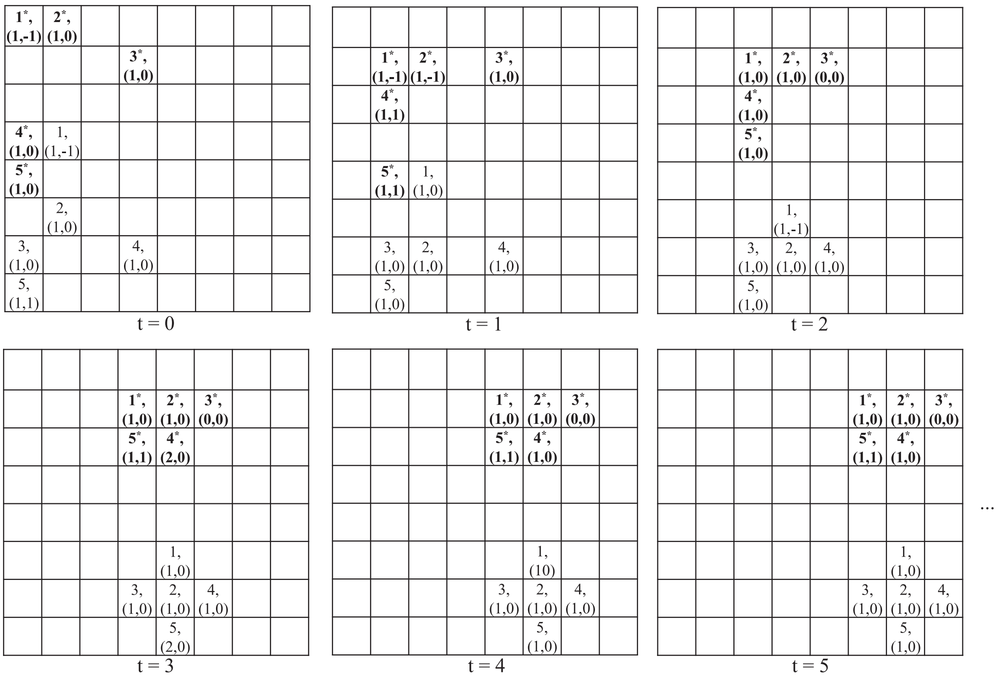

Fig. 2. Snapshot of emergent property states.

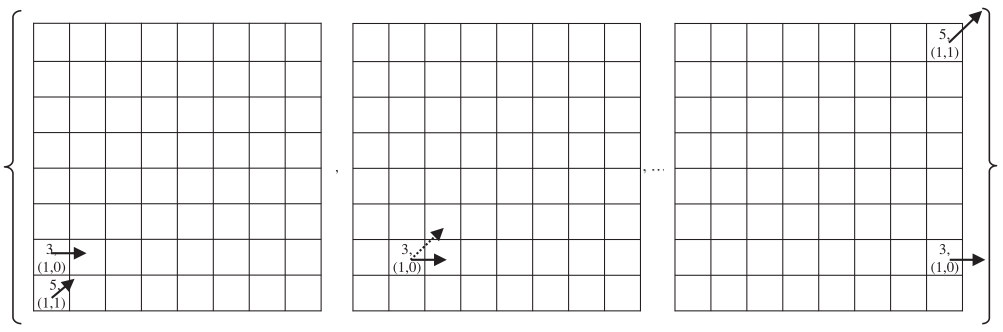

Fig. 3. Example of $L(A_{23}) \oplus (L(A_{25}))$.

4.2. Emergence Identification

We show a snapshot of how the flocking of birds, a well-known emergent property, is detected. The initial system state is given as the state at time t = 0 in Figure 2. For ease of visualization, we distinguish ducks from geese using a star ($*$) symbol and bolded cell. $< j, (\alpha, \beta) >$ denotes a bird instance j, with velocity (α,β). Figure 2 shows a simulation of the system when $\epsilon = 2$, $\lambda = 10$, and $\beta = 8$. We observe that the birds keep flying in the same pattern since t = 4. Moreover, the system gets back to the system state $S(4)$ at time t = 12: $S(12) = S(4)$. Hence, $L^{I}_{whole} = \{S(0), S(1), \ldots, S(11)\}$.

L_{sum} is calculated using the superimposition operator as $L_{sum} = \oplus(L(A_{11}), \ldots, L(A_{25}))$. By definition and following our previous discussion, L_{sum} tends to be very large, even for small problem sizes. As such, we consider for illustration two geese A_{23} and A_{25}. The visualized result of the superimposition of $L(A_{23})$ and $L(A_{25})$ is shown in Figure 3.

Table III. Sizes of L^I_{whole}, L_{sum}, and L_ξ

Number of Birds	Number of States			$\frac{L_\xi}{L^I_{whole}}$
	L^I_{whole}	L_{sum}	L_ξ	
4	13	767	6	0.46
6	18	70,118	12	0.67
8	13	509,103	9	0.69
10	26	13,314,066	23	0.88

Formally,

$$\begin{aligned}
&\oplus (L(A_{23}), L(A_{25})) \\
&= L(A_{23}) \oplus (L(A_{25})) \cup L(A_{25}) \oplus (L(A_{23})) \\
&= \{(fff \ldots (2,(1,7),(1,0)) \ldots (2,(1,8),(1,1)) \ldots), \\
&\quad (fff \ldots (2,(2,7),(1,0)) \ldots), \\
&\quad (fff \ldots (2,(8,1),(1,1)) \ldots (2,(8,7),(1,0)) \ldots), \ldots\}
\end{aligned}$$

$L(A_{23})$ and $L(A_{25})$ are behaviors of agent A_{23} and A_{25}. f represents an empty cell and a tuple $< i, (x, y), (\alpha, \beta) >$ represents the state of a bird of type i at cell (x, y) with velocity (α, β). For example, $< 2, (1, 7), (1, 0) >$ represents a goose locating at cell $(1, 7)$ with velocity $(1, 0)$. Note that the x-axis is horizontal and oriented from left to right, and the y-axis is vertical and oriented from top to bottom. For the 10-bird example, we can run the simulation to determine L_{sum}, and see that $L_\xi = \{S(1), S(2), \ldots, S(11)\}$. Furthermore, we assume that a group of birds flock if at least five birds of the same type fly *together*, with each bird having at least one immediate neighbor of the same type. As a result, flocking is a known emergent property. Ten emergent property states from $S(2)$ to $S(11)$ therefore have flocking emergent property.

4.3. Experimental Analysis

In this section, we present the experimental analysis of our approach. Since we derived the sets of states for L^I_{whole}, L_{sum}, and L_ξ, we present the complexity in terms of the number of states. We implemented our approach in Java, and our experimental analysis quantifies the state-space size by varying the number of birds.

4.3.1. Experimental Results: State-Space Explosion. We computed and analyzed the states that make up L^I_{whole}, L_{sum}, and L_ξ. We also analyzed how agent interactions affect the size of L_ξ with respect to the size of L^I_{whole}. As L_{sum} suffers from state-space explosion, we varied the number of birds from four to 10, with equal number of ducks and geese. A difference between ducks and geese is their maximum speed, which is two cells per time step for ducks and three cells per time step for geese. The location and velocity of all birds are initialized randomly. Both ducks and geese follow the three behavior rules defined in Section 4. Given an initial system state, at some point of time t, the system will arrive at a state that has already happened at time $t' < t$ because the number of possible system states is finite, even if it can be very large. The size of L^I_{whole} is the number of distinct system states obtained from the beginning until time t. L_{sum} is computed over all possible combinations of individual agents' behaviors with respect to the initial system state. The initial system state belongs to both L^I_{whole} and L_{sum}.

For every experiment, we ran the simulation 10 times and took the averages where applicable as shown in Table III. The experiments are run using a 2.4GHz machine with 3GB RAM. As expected, the size of L_{sum} is large and increases exponentially with the number of birds. For instance, L_{sum} grows by 90 times when the number of birds changes from four to six. Another interesting observation is that L_ξ / L^I_{whole}

Table IV. Emergent Property Set Size for a 128 × 128 Grid with $\delta = 0.1$

Number of Birds	L^I_{whole}	L_o None	L_o Weak	L_ξ	Execution Time (s)
4	133	17	21	95	small
8	157	3	19	135	small
16	643	5	16	622	0.2
32	1,158	2	19	1,137	1.3
64	3,037	1	15	3,021	12.8
128	7,497	1	12	7,484	151.7
256	6,871	1	13	6,857	941.8
512	5,038	1	11	5,026	5,394.3
1,024	4,072	1	11	4,060	32,266.8

tends to increase with the number of birds. In other words, more interactions between birds lead to more emergent property states that cannot be derived by summing the independent agents' behaviors. Furthermore, the number of common elements between L^I_{whole} and L_{sum} is small. Consequently, computation is wasted on calculating L^{NP}_{sum} states that are not feasible in practice. However, when the impact of neighboring agents on an agent is not strong, then L^{NP}_{sum} tends to be small. For example, agents may not interact frequently because they are far from each other, such as when the number of agents is much smaller than the number of cells in the environment.

4.3.2. Experimental Results: State-Space Reduction. By eliminating L_{sum} in this method as discussed in Section 3.4, our experiments scale up to a larger number of birds (1,024 birds) and a larger environment (128 × 128 grid). We set δ to 0.1 to consider that emergence occurs when the degree of interaction between birds is larger than 0.1. Other values of δ are examined later. The bird position is the only attribute considered in the calculation of the degree of interaction, as position is the result of applying all behavior rules, and thus changes in position demonstrate how strong birds' interactions are. Table IV shows experimental results for different numbers of birds on a 128 × 128 grid. We keep the population of ducks and geese equal. The purpose of the experiments is to analyze the relationships among L^I_{whole}, L_o, and L_ξ. We are also interested in evaluating the scalability of the new method of determining emergence.

The first observation is that when the number of birds doubles, L^I_{whole} grows as expected, but soon drops. For example, for a 128 × 128 grid, L^I_{whole} decreases sharply from 7,497 to 4,072 when the number of birds increases from 128 to 1,024 (see Table IV). Similarly, L^I_{whole} changes noticeably from 3,803 to 1,536 when the bird population varies from 128 to 1,024 in a 64 × 64 grid. This is probably because a larger number of birds encourages more interactions, thus making the birds' movement more structured.

Second, for all explored environment sizes, the system states due to weak interaction are small and tend to decrease with the number of birds. This further clarifies the close relationship between the size of the bird population and the degree of interaction, in that more birds lead to more interactions, hence smaller L_o. Furthermore, the number of system states due to no interactions tends to reach one, the initial state.

Third, L_o is small compared to L_ξ, especially when the number of birds is large. For example, $\frac{L_o}{L_\xi}$ is less than 1%. There are two possible reasons for this. The first reason is that $\delta = 0.1$ is a small value. If δ is set to a larger value, we are likely to retrieve fewer emergent property states but with a higher degree of interactions. However, there is a risk that some appealing emergent properties may be ignored. The second reason is that interactions between birds when the group becomes more crowded amplify the

Table V. Size of L^I_{whole} and L_ξ in a 16 × 16 Grid for Different Values of δ

Number of Birds	L^I_{whole}	L_ξ					
		0.0	0.1	0.2	0.3	0.4	0.5
4	18	3	3	6	5	1	0
8	44	3	4	21	13	3	0
16	65	2	2	22	33	6	0
32	127	2	2	20	93	10	0
64	220	2	1	31	183	3	0

degree of interaction, thus making the degree larger than δ for most system states. This needs to be further investigated.

Our analysis of the execution times shows that the new proposed method of identifying emergent property states is much more efficient: eliminating L_{sum} enables experiments of much larger bird populations and environment sizes (1,024 birds and a 128 × 128 grid in the new method compared to 10 birds and an 8 × 8 grid in the original method). In particular, for 1,024 birds and a 128 × 128 grid, the experiment took about 9 hours to run through 4,072 system states and classify them into nonemergent and emergent property states based on the degree of interaction. Compared to the original approach of calculation of L_ξ, which took about 1 hour for only 10 birds and an 8 × 8 grid, we can see the high efficiency of the proposed technique for state-space reduction.

To understand the relationship between L_ξ and δ, we perform experiments for different numbers of birds and different values of δ in a 16 × 16 grid as shown in Table V.

We observe that L_ξ has the largest number of states when the degree of interaction is between [0.2, 0.3] and [0.3, 0.4]. Considering 64 birds, these two ranges account for about 97% of emergent property states. In addition, [0.3, 0.4] seems to have more emergent property states than [0.2, 0.3], when the bird population increases. For instance, the ratio of L_ξ in [0.3, 0.4] and [0.2, 0.3] grows from 0.8 to 5.9 when the number of birds grows from four to 64, respectively. Our hypothesis is that behavior rules follow a particular distribution of degree of birds' interaction, but this has to be further investigated.

4.4. Discussion

The idea of using the degree of interaction has a limitation in that the degree of interaction depends on the attributes that are selected to calculate it. Our experiments show that it is not always straightforward to select attributes such that the degree of interaction covers the whole range of [0,1], in which zero means no or cancelled out interaction and one refers to possibly maximum interaction. Through the use of the degree of interaction, the computationally expensive calculation of L_{sum} is avoided and thus the runtime of our approach is significantly reduced. The use of the degree of interaction makes our approach less generic than that using L_{sum} by relying on the insight that interaction accompanies emergent behavior. However, it also makes our approach more practical by making it computationally feasible: to ensure that this approach is universally applicable, further experimentation is required.

Our formalism and approach allow us to identify the states that form L_ξ, the set of emergent property states. These states are manifestations or instances of emergent properties. For example, the emergent property of "flocking" can have more than one manifestation or instance, in which birds are clustered around different virtual centers on a grid. The identification of a specific emergent property from an emergent property state set requires the intervention of the system expert. Toward this, the system expert that analyzes L_ξ could define some system-specific emergence criterion. For example, a flocking criterion requiring a minimum number of birds to cluster around a point can be used to highlight a number of states from L_ξ as belonging to the emergent property state of "flocking." Alternatively, the set could be compared with sets extracted from systems that have been previously shown to exhibit emergence, similar to the approach proposed in Birdsey and Szabo [2014].

Next, to increase the adoption of our approach, the multiagent system under study could be specified in any formalism, including the DEVS-based formalism proposed by Mittal [2013]. Toward this, transformation functions that preserve semantics and logical properties need to be defined and validated. For some constructs, these can be straightforward mappings, for example, agent to agent capturing attributes, whereas for others, the transformations are not straightforward, for example, the DEVS behavior functions to the rule-based definition and grammar proposed by our approach. Lastly, our approach needs to be further validated by applying it to other systems. Our experiments also include the Game of Life and traffic jams in small intersections; predator–prey and social networks will be considered as part of our future work.

5. CONCLUSION

This article proposes a formalization of emergent behavior using an extended cooperating array grammar system to identify the existence and the size of emergent property state sets in multiagent systems. In weak emergence, the set of emergent property states for a given system (L_ξ) is derived by taking the difference between the set of observed system states due to agent interactions (L_{whole}) and the set of system states obtained by combining the behaviors of individual agents (L_{sum}). Our formalism models multiagent systems with agents of various types and characteristics and permits the modeling of open systems. Our formalization allows for the automated calculation of L_ξ. However, our theoretical analysis shows that the formalism suffers from severe state-space explosion. This is also shown by our experimental analysis, which studies a flock-of-birds model with two types of birds but only allows us to analyze models with up to 10 birds. We address this drawback by introducing a degree-of-interaction metric that significantly reduces the state space. The formalization and subsequent implementation of the degree-of-interaction metric permit us to analyze models with up to 1,024 birds on significantly larger two-dimensional grids (128×128).

Our approach is a significant improvement over existing work, both through considering agents with more than one attribute, mobile agents, and open systems and through its automation potential, which has a reduced runtime through the use of the degree-of-interaction metric. However, the degree of interaction is applicable only to systems where emergent behavior is a result of direct interaction and does not consider indirect interactions between agents. This is a tradeoff between computational cost and applicability that needs to be further analyzed. This article provides a first step toward advancing our understanding of emergence and introduces a scalable, automated way for identifying emergent property states, but much work remains to be done. More efficient state representations, such as bit state hashing and state vector compression used in model checking, are required. We have applied this approach to the Game of Life and to traffic jam models and are also exploring the application of this approach to study emergence in social networks and concurrent program specification. Given a program and its specification, the cause and effect of properties such as deadlock can be analyzed by examining the corresponding system states.

ACKNOWLEDGMENTS

The authors wish to thank Luong Ba Linh for discussions about this work.

REFERENCES

R. Abbott. 2006. Emergence explained: Abstractions getting epiphenomena to do real work. *Complexity* 12, 1, 13–26.

L. A. Adamic and B. A. Huberman. 2000. Power-law distribution of the World Wide Web. *Science* 287, 2115.

R. Albert, H. Jeong, and A. L. Barabasi. 1999. Diameter of the World Wide Web. *Nature* 401, 130–131.

N. A. Baas and C. Emmeche. 1997. On emergence and explanation. *Intellectica* 2, 67–83.

Y. Bar-Yam. 2004. A mathematical theory of strong emergence using multiscale variety. *Complexity* 9, 6, 15–24.

M. A. Bedau. 1997. Philosophical perspectives: Mind, causation and world. *Philosophical Perspectives Annual Volume* 11.

M. A. Bedau. 2003. Downward causation and the autonomy of weak emergence. *Principia 3* 3, 5–50.

C. Bernon, M.-P. Gleizes, S. Peyruqueou, and G. Picard. 2003. Adelfe: A methodology for adaptive multi-agent systems engineering. In *Engineering Societies in the Agents World III*. Springer, 156–169.

L. Birdsey and C. Szabo. 2014. An architecture for identifying emergent behavior in multi-agent systems. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems*. 1455–1456.

E. Bonabeau and J. Dessalles. 1997. Detection and emergence. *Intellectica* 25, 2.

D. J. Chalmers. 2006. Strong and weak emergence. In *The Re-emergence of Emergence*. Oxford University Press.

W. K. V. Chan. 2010. Interaction metric of emergent behaviors in agent-based simulation. In *Proceedings of Winter Simulation Conference*, S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu (Eds.). Institute of Electrical and Electronics Engineers, 357–336.

C. C. Chen, S. B. Nagl, and C. D. Clack. 2009a. Complexity and emergence in engineering systems. *Complex Systems in Knowledge-based Environments: Theory, Models and Applications* 168, 99–128.

C. C. Chen, S. B. Nagl, and C. D. Clack. 2009b. A formalism for multi-level emergent behaviours in designed component-based systems and agent-based simulations. In *From System Complexity to Emergent Properties*. Springer-Verlag, 101–114.

L. Chi. 2009. Translating social capital to the online world: Insights from two experimental studies. *Journal of Organizational Computing and Electronic Commerce* 19, 214–236.

N. Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2, 113–124.

J. P. Crutchfield. 1999. Is anything ever new? Considering emergence. In *Complexity*, Vol. 19, 515–537. Addison-Wesley.

V. Darley. 1994. Emergent phenomena and complexity. *Artificial Life IV*, 4, 411–416.

J. Deguet, L. Magnin, and Y. Demazeau. 2006. Elements about the emergence issue: A survey of emergence definitions. *ComPlexUs* 3, 24–31.

G. B. Dyson. 1998. *Darwin Among the Machines: The Evolution of Global Intelligence*. Perseus Books Group.

D. Fisch, M. Janicke, B. Sick, and C. Muller-Schloer. 2010. Quantitative emergence - A refined approach based on divergence measures. In *Proceedings of 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. 94–103.

S. Floyd and V. Jacobson. 1994. The synchronization of periodic routing messages. *IEEE/ACM Transactions on Networking* 2, 2, 122–136.

J. Fromm. 2007. Types and Forms of Emergence. http://arxiv.org/abs/nlin.AO/0506028.

J. M. E. Gabbai, H. Yin, W. A. Wright, and N. M. Allinson. 2005. Self-organization, emergence and multi-agent systems. In *Proceedings of International Conference on Neural Networks and Brain*. 1858–1863.

R. Gore and P. F. Reynolds. 2008. Applying causal inference to understand emergent behavior. In *Proceedings of the Winter Simulation Conference*. 712–721.

P. Haglich, C. Rouff, and L. Pullum. 2010. Detecting emergence in social networks. In *Proceedings of IEEE 2nd International Conference on Social Computing*. 693–696.

B. Heath, R. Hill, and F. Ciarallo. 2009. A survey of agent-based modeling practices (January 1998 to July 2008). *Journal of Artificial Societies and Social Simulation* 12, 4, 9+.

J. H. Holland. 1997. *Emergence: From Chaos To Order*. Addison Wesley.

R. Holzer, H. De Meer, and C. Bettstetter. 2008. On autonomy and emergence in self-organizing systems. In *Proceedings of 3rd International Workshop on Self-Organizing Systems*. 157–169.

J. Hopcroft, R. Motwani, and J. Ullman. 2001. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

P. Hovda. 2008. Quantifying weak emergence. *Journal of Minds and Machine* 18, 4, 461–473.

M. Jacyno, S. Bullock, M. Luck, and T. R. Payne. 2009. Emergent service provisioning and demand estimation through self-organizing agent communities. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. 481–488.

C. W. Johnson. 2006. What are emergent properties and how do they affect the engineering of complex systems? *Reliability Engineering and System Safety* 12, 1475–1481.

A. Kubik. 2003. Towards a formalization of emergence. *Journal of Artificial Life* 9, 1, 41–65.

Z. Li, C. H. Sim, and M. Y. H. Low. 2006. A survey of emergent behaviour and impacts in agent-based systems. In *Proceedings of IEEE International Conference on Industrial Economics*. 1295–1300.

E. J. Manley and T. Cheng. 2010. Understanding road congestion as an emergent property of traffic networks. In *Proceedings of 14th World Multi-Conference on Systemics, Cybernetics and Informatics*. 25–34.

F. S. Melo and M. Veloso. 2009. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. 773–780.

S. Mittal. 2013. Emergence in stigmergic and complex adaptive systems: A formal discrete event systems perspective. *Cognitive Systems Research* 21, 22–39.

M. Mnif and C. Muller-Schloer. 2006. Quantitative emergence. In *Proceedings of IEEE Mountain Workshop on Adaptive and Learning Systems*. 78–84.

J. C. Mogul. 2006. Emergent (mis)behavior vs. complex software systems. In *Proceedings of 1st ACM SIGOPS/EuroSys European Conference on Computer Systems*. 293–304.

T. Moncion, P. Amar, and G. Hutzler. 2010. Automatic characterization of emergent phenomena in complex systems. *Journal of Biological Physics and Chemistry* 10, 16–23.

I. Norros, B. J. Prabhu, and H. Reittu. 2006. Flash crowd in a file sharing system based on random encounters. In *Proceedings of Workshop on Interdisciplinary Systems Approach in Performance Evaluation and Design of Computer & Communications Systems*. 42–51.

T. O'Conner. 1994. Emergent properties. *American Philosophical Quarterly* 31, 2, 91–104.

K. K. Ramakrishnan and H. Yang. 1994. The Ethernet capture effect: Analysis and solution. In *Proceedings of 19th Conference on Local Computer Networks*. 228–240.

M. Randles, H. Zhu, and A. Taleb-Bendiab. 2007. A formal approach to the engineering of emergence and its recurrence. In *Proceedings of 2nd International Workshop on Engineering Emergence in Decentralized Autonomic Systems*. 1–10.

C. W. Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of 14th Annual Conference on Computer Graphics and Interactive Techniques*. 25–34.

C. Rouff, A. Vanderbilt, M. Hinchey, W. Truszkowski, and J. Rash. 2004. Properties of a formal method for prediction of emergent behaviors in swarm-based systems. In *Proceedings of 2nd IEEE International Conference on Software Engineering and Formal Methods*. 24–33.

N. Salazar Salazar, J. Rodriguez-Aguilar, J. Arcos, A. Peleteiro, and J. Burguillo-Rial. 2011. Emerging cooperation on complex networks. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. 669–676.

A. K. Seth. 2008. Measuring emergence via nonlinear Granger causality. *Artificial Life XI* 324, 1, 545–552.

C. E. Shannon. 2000. A mathematical theory of communication. *Bell System Technical Journal* 27, 3, 379–423.

C. Szabo and Y. M. Teo. 2012a. An integrated approach for the validation of emergence in component-based simulation models. In *Proceedings of the Winter Simulation Conference*. 2412–2423.

C. Szabo and Y. M. Teo. 2012b. Semantic validation of emergent properties in component-based simulation models. *Ontology, Epistemology, and Teleology of Modeling and Simulation Philosophical Foundations for Intelligent M&S Applications*. 319–333.

Y. M. Teo, L. B. Linh, and C. Szabo. 2013. Formalization of emergence in multi-agent systems. In *Proceedings of the ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. 231–240.

T. De Wolf, G. Samaey, T. Holvoet, and D. Roose. 2005. De-centralized autonomic computing: Analysing self-organizing emergent behavior using advanced numerical methods. In *Proceedings of the 2nd International Conference on Autonomic Computing*. 52–63.

B. Zhan, D. N. Monekosso, P. Remagnino, S. A. Velastin, and L. Q. Xu. 2008. Crowd analysis: A survey. *Machine Vision and Application* 19, 5–6, 345–357.

Received February 2014; revised May 2015; accepted July 2015