2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation

An Objective-based Approach for Semantic Validation of Emergence in Component-based Simulation Models

Claudia Szabo
Department of Computer Science
The University of Adelaide
North Terrace
Adelaide 5005
claudia.szabo@adelaide.edu.au

Yong Meng Teo
Department of Computer Science
National University of Singapore
Computing 1, 13 Computing Drive
Singapore 117417
teoym@comp.nus.edu.sg

***Abstract*—Component-based models have been shown to exhibit emergent properties but despite a plethora of definitions and methods to identify emergence, practical semantic validation approaches remain a key challenge. This paper proposes an objective-based approach for semantic validation of emergence in component-based simulation models. In contrast to current methods, our approach describes model components in terms of what they achieve rather than how they achieve it, and we exploit reconstructability analysis in the validation of emergence. This has the advantage of reducing the number of attributes used to describe each component, and thus facilitates the application of more rigorous mathematical formalisms for emergence validation. As an example, we detail how this methodology is integrated into the life-cycle of our component-based model development framework.**

I. Introduction

Complex systems often exhibit properties that are not easily predictable by analyzing the behavior of their individual, interacting components [7], [14], [17]. These properties, called emergent properties, are increasingly becoming important as software systems grow in complexity, coupling, and geographic distribution [1], [13], [14], [17]. Examples of emergent properties include connection patterns in data extracted from social networks [6], trends in big data analytics [8], and power supply variation in smart grids due to provider competition [3]. More malign examples of emergent properties in computer systems include the Ethernet capture effect [20], router synchronization problems [9], and load-balancer failures in a multi-tiered distributed system [17]. Because emergent properties may have undesired and unpredictable effects and consequences, and unpredictable systems are less credible and difficult to manage, techniques for the identification and validation of emergent properties are becoming of crucial importance. Despite ongoing research interest since the 1970s, most approaches focus only on the post-mortem observation of emergence in various biological, social, and AI contexts, and less on measuring and advancing our understanding in the cause-and-effect of emergence. A plethora of examples of emergent properties have been identified and classified but few have been measured and explained [5], [13], [16], [17].

In modeling and simulation, the possibility of emergent behavior in component-based model development has been highlighted since the 1990's by Page and Opper [18], which propose a formal framework for analyzing the complexity of composition and emergence. A more practical perspective proposes to look at the simulation code and highlight the lines that generate a particular variable value [12]. Despite ongoing research [4], [11], [12], [18], the validation of emergence remains an important issue. An important challenge is the need to abstract the component-based simulation model at two levels, namely, the level of individual components or *micro-level*, and the level of the composed model as a whole, or *macro-level*. This abstraction is difficult to achieve in an automated manner, and hence most approaches rely on a post-mortem observation of the composed model simulation by a system expert [12], instead of validating emergence without a-priori knowledge [16]. Moreover, current approaches are applicable to specific simplified models [16] and are not easily applied to other models of increased scale or complexity [25]. There is a need for a formal approach for the validation of emergent properties that can be applied to a variety of component-based models, both in terms of size and complexity, to increase the confidence in the composed model [1], [16].

In this paper, we propose an approach for the semantic validation of emergent properties in component-based simulation models. Our two-step approach first identifies simulation states that are candidates for emergence using an objective-based specification of the model components. Next, towards the semantic validation of emergence, we propose to highlight the simulation states leading to and from the emergence candidates. Our approach relies on the specification of a model component as a meta-component that includes among others a definition of the objective that the sub-component achieves. In the first step, we compute the entire composed model state and compare it with an objective-based reconstruction of the system state. The states with a significant difference are added to an *emergence set*. In the second step, we propose

978-0-7695-4714-5/12 $26.00 © 2012 IEEE
DOI 10.1109/PADS.2012.9

to highlight the states leading to and from emergence for a better understanding of the composed model. We represent the simulation execution as a Labeled Transition System (LTS) [23] and analyze the semantic difference between the states in the emergence set and the other composed model states. The semantic difference is calculated using attributes and properties represented in our proposed component-based simulation ontology. The contributions of this paper are twofold. Firstly, we propose an objective-based emergence validation approach that does not rely on an a-priori identification of emergence. Our novel approach uses reconstructability analysis [2] to construct a composed model state from the specifications of the individual model components and highlight candidate emergent states. Secondly, for a better understanding of the composed model, our semantic validation identifies the differences between states in the composed model simulation to highlight situations leading to and from emergence.

This paper is organized as follows. Section II presents an overview of emergence concepts, and discusses current work in validating emergence in complex systems and in component-based modeling and simulation. We discuss our objective-based proposed approach in Section III. Section IV concludes this paper and presents future work.

II. Related Work

An emergent property can be defined as "a property of an assemblage that could not be predicted by examining the components individually" [1]. Common characteristics of emergence include: radical novelty (features not previously observed in systems); coherence or correlation (meaning integrated wholes that maintain themselves over some period of time); a global or macro "level" (i.e. there is some property of "wholeness"); it is the product of a dynamical process (it evolves); and it is "ostensive" (it can be perceived). The fundamentals behind understanding different types of emergence lie in the assumption that in any complex system there is a *micro-level*, the abstraction level of each individual component, and a *macro-level*, the abstraction of the composed model as a whole. The micro-level properties are usually measured by observing the component states, e.g. the collection of all variables and their values, of each system component. In contrast, the macro-level properties can be measured either as an aggregation of all the states of the system sub-components, or by observing the overall system behavior and observing trends. For example, in a model of a flock of birds a change could be from non-flocking to flocking behavior.

Three main types of emergence have been identified, namely *nominal*, *strong*, and *weak* [1]. In *nominal emergence*, the macro-levels depend on the micro-levels in the straightforward sense that wholes are dependent on their constituents. *Strong emergence* is a more powerful definition that assumes nominal emergence, but introduces *downward causation*, which can be informally defined as the influence of the macro-level on the micro-levels. In contrast, *weak emergence* states that given the properties of the parts and the interaction rules among them, it is not *trivial* to infer the properties of the whole. In this context, trivial is taken to mean "by-hand" human calculations. That is, in order to identify weak emergence, one needs a computer model and its simulation.

In the complex systems domain, emergence validation approaches can be classified in three main categories, namely, *grammar-based*, *variable-based*, and *event-based*. *Grammar-based methods* consist of two grammars, L_{WHOLE} to describe the properties of the system as a whole, and L_{PARTS} to describe the properties obtained from the reunion of the parts [16]. Kubik [16] proposes the use of grammar systems, which are symbolic devices composed from a set of grammars that interact with each other through tapes on which each grammar writes symbols. A formal *grammar* is a set of rules that governs the formation of *words* using a set of *symbols*. This paradigm applies easily to multi-agent systems, where each agent can be represented by a grammar and the behavior of an agent is represented by how it changes the symbols on the common tape. Emergence is defined as the difference between the properties of the system as a whole L_{WHOLE}, and the reunion of the properties of the system parts, L_{PARTS} [16]. To calculate L_{PARTS}, Kubik [16] proposes the superimposition of each agent language defined by the grammar system. Informally, L_{PARTS} is defined by the sum of the changes or conditions the agents bring about the environment if they would act individually in the system. In this superimposition, L_{PARTS} is formed using a reunion operator for all possible permutations of words created following rules that give higher priority to the symbols generated by the agent grammar, and less priority to the system symbols. While L_{PARTS} uses the superimposition operator to highlight the behavior of agents without considering the agent interaction with the environment, L_{WHOLE} is obtained as a reunion of all the symbols generated by agents. This method does not require a prior observation of the system to identify possible emergent properties or behaviors, which makes it suitable for large-scale composed models where such observations are almost impossible. However, the nature of the formalism and the computation of the composed model states make it difficult to scale.

In *variable-based methods*, a specific variable is chosen to describe emergence. Changes in the values of this variable are said to signify the presence of emergence properties [22]. For example, the centre of mass of a bird flock could be used as an example of emergence in bird flocking behavior, as shown in [22]. The approach uses Granger causality to establish the relationships between a macro-variable, representing a system property, and micro-variables, representing properties of the system sub-components. This approach provides a clear and easily measurable process to identify emergence because it looks at measurable quantities found in the system state, which is defined as the reunion of all sub-systems states. However, finding a good variable to describe a system can be a difficult task that requires system expert intervention and extensive observations of the system. Moreover, Granger causality handles only pairs of variables, and might not apply when the macro-variable depends on more than one micro-

variable.

In *event-based methods*, behavior is defined as a series of events which change a system or a sub-system state [5]. The motivating example behind this work is that often when a macro-level property is constructed from the aggregation of component states, there is a loss of information with respect to the *cause* of the emergent behavior. In particular, it is not possible to establish which component interaction is responsible for the current behavior. Towards this, the authors propose the definition of simple and complex event types. A *simple event* type signifies a change in a sub-system state. It is associated with a transition and has a duration. A *complex event* is defined as being either a simple event or constituted from two complex events linked by a relationship. This relationship is a temporal operator (meaning that there is a temporal relationship between the two complex events) that can optionally have descriptions of constraints related to the environment or to the state of the two sub-systems.

The approaches described above can be categorized into *a-priori* and *a-posteriori* methods. In *a-priori* methods such as the variable-based and event-based approaches, there needs to be a variable or a complex event that defines the emergent property, respectively. The identification of this variable is manual and might not be straightforward for more complex composed models. In *a-posteriori* methods such as the grammar-based approaches, a formalism guides the identification of emergence as a difference between the outcome of the interaction between the components in the system, and the outcome calculated if no interaction between components occurs. Ideally, the latter approach would be suitable for complex composed models where a single variable to define emergence is difficult to find. Our previous study [25] has shown that both a-priori and a-posteriori methods can be easily applied to specific classes of composed models, but are difficult to apply on more complex composed models. The limiting factor is the formalism employed, which needs to cater for both high and low levels of abstraction.

The possibility of emergent behavior in component-based simulation model development has been highlighted since the 1990's by Page and Opper [18], which propose a formal framework for analyzing the complexity of composition and emergence. They propose a slightly different definition of emergent behavior than the systems theory, using the given components a and b that are composed as $a \diamond b$, an objective o, and the "satisfies" operator $\models$. If $a \models o$, then a satisfies objective o. If $a \not\models o$ and $b \not\models o$, but $(a \diamond b) \models o$, then we can say that the composition is *emergent*. In a similar manner to variable-based methods, Gore and Reynolds denote emergence as a specific variable value and propose to highlight the lines in the simulation source code that cause that particular value [12]. They further propose a taxonomy for analyzing emergent behavior based on reproducibility, predictability, and temporality [11]. Reproducibiliy refers to the repeatability of a simulation for a given set of inputs. Predictable behaviors enable selective sampling towards testing user hypotheses. Temporality distinguishes between the simulation reaching a final state and residing in the final state. The proposed taxonomy allows system experts to classify a particular emergent behavior. However, the identification of emergent behavior as well as its validation are not addressed. Another approach to identify emergence proposes to look at a measure of the interaction between agents in an agent-based model [4]. The interaction metric is an agent-specific counter that increases as the agent interacts with other agents in the environment. Emergence is said to appear if the interaction measure deviates from normality. This approach provides a straightforward measure of emergence. However, the study considers only simple models such as Conway's Game of Life [10] and the flock of birds model [21]. Moreover, cases where emergence is a result of indirect interaction between agents are not addressed.

III. Proposed Approach

In our previous work, we proposed a component-based simulation model development framework called CoDES that proposed a semantic validation of expected model behavior [26]. In this section, we extend this framework for semantic validation of emergent behavior. We present the required extension in our component abstraction model and ontology to support a new objective-based approach for semantic validation of emergent properties in newly composed simulation models.

A. Extension of Component Abstraction and Ontology for Emergence

Our framework consists of four main steps, namely *conceptual model definition*, *syntactic verification*, *model discovery and selection*, and *semantic validation*. In *Conceptual Model Definition*, a simulation problem is translated into a conceptual model using a graphical editor with component icons, denoting model component stubs, interconnected using connectors. *Syntactic Verification* of the structure of the conceptual model is automated using EBNF composition grammar [26]. Model components in a syntactically correct conceptual model are discovered in *Model Discovery and Selection*. Next, attributes and behavior are instantiated in the composed model, and semantic validation is performed. In the remaining, we employ an example of a simple model developed using model components from the Queueing Networks application domain. Examples of model component in a Queueing Networks are source, server, and sink, as shown in Figure 4. A *Source* base component generates jobs at intervals and passes the job to the next component. A *server* component encapsulates one or more service units and is served by a single queue. Completed jobs terminate at the *sink*. In our previous work, we validate expected behavior that arises from the interactions of the underlying components. In this paper, we focus on unexpected or emergence behavior arising from seemingly unrelated phenomena.

Model components in a composed model are abstracted as *meta-components* and a component C_i is represented as:

$$C_i = \langle R, A_i, B_i \rangle$$

where R denotes required attributes, that are common to all components and generally employed for version control, e.g. `author, location, lastUsed`, A_i denotes component specific attributes, e.g. `interArrivalTime`, `numJobsServiced`, and B_i represents component behavior as a state machine. A component behavior is represented as:

$$[I_l]S_p[\Delta t] \xrightarrow{Cond_n} S_t[O_l][A_m]$$

where I_l: the set of input data; S_p: the current state; Δt: the transition duration; $Cond_m$: the condition(s) for the state transition; S_t: the next state; O_l: the set of outputs; A_m: the set of modified attributes after the state change. The strings in A_i and B_i are defined in our proposed COSMO component-based ontology, as described below.

To support objective-based emergent properties validation, each component C_i has an added objective $o(C_i)$ that describes *what* the component achieves. For example, an objective for a Source model component in a queueing networks model would be "Objective 1: generates jobs every 3.5 seconds on average". The objective is defined using both textual description and a set of variables, $var(o(C_i))$, with a type and a range of values that signify the objective is met. More formally,

$$var = \{(type, values) | type \in COSMO\}$$

where the variable type is defined in COSMO, and can be of primitive types such as `int` or more complex types such as `JobClass`. The variables also represent a subset of the attributes defined in the state machine. Thus,

$$o(C_i) = \{(description, var) | var \in A_m\}$$

As such, the variables defining the component objective can change value with transitions. When these variables reach values in the specified range, the objective is said to be met. For the *Source* component C_1 in Figure 4, the objective can be defined as

$$o(C_1) = (description, var(o(C_1)))$$

where

$$description = Generates\ jobs\ every\ 3.5\ seconds\ on\ average$$

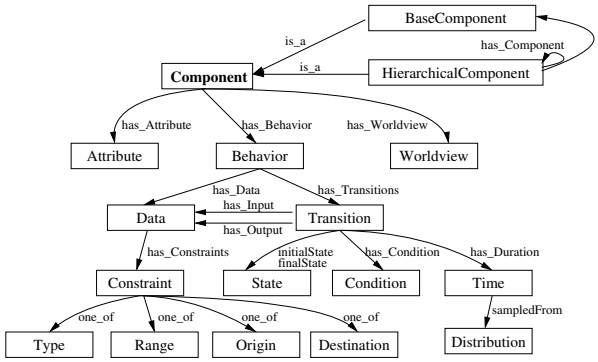
and

$$var(o(C_1)) = \{numJobsGenerated, interArrivalTime\}$$

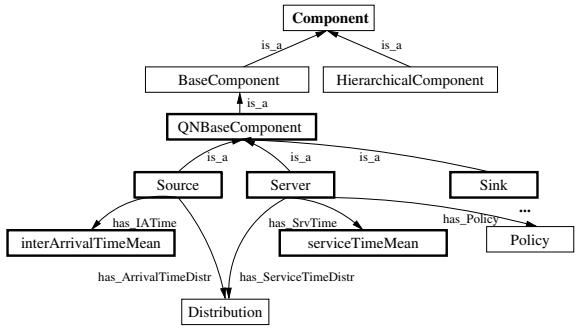
The attribute *numJobsGenerated* captures the fact that the *Source* component generates jobs, whereas the *interArrivalTime* captures the second part of the objective.

This objective-based definition of a component captures the relevant attributes that define the component behavior. This reduces the number of attributes considered in emergence validation and permits the application of reconstructability analysis. However, several assumptions are in place. Firstly, the objective must be defined as specified above, by the model component creator. Secondly, the variables defining the objective must be already part of the component definition, as specified by $var \in A_m$. A higher level representation should have these objectives defined as independent of the component attributes, with additional rules specifying when these objectives are met. This approach also has its disadvantages and its study is part of our future work.

Capturing domain specific knowledge is of crucial importance to understand complex models [24]. We propose the COSMO ontology for describing component-oriented simulation within and across application domains to support discovery, reuse, and validation of the composed models [26]. The COSMO hierarchy spans two main directions, as shown in Figure 1(a). To achieve generality across domains



(a) COSMO Ontology Structure



(b) COSMO Ontology for Queueing Networks

Fig. 1. Ontology for Component-based Simulation Development

and at the same time support specific domain requirements, we include a model component oriented hierarchy of base and hierarchical components, followed by a domain-oriented hierarchy. The first level of COSMO describes components with respect to their *attributes* and *behavior*, as discussed in Section III-A. The classes for attribute, behavior, worldview, transition, state etc., are defined in the ontology. The addition of the Queueing Networks domain to CoDES is reflected in the second level of the COSMO ontology as shown in Figure 1(b). *QNBaseComponent* is added as a new subclass of BaseComponent, with *Source*, *Server*, and *Sink* as its subclasses. Besides having attributes and behavior like their superclass, the Source component must have an *interArrivalTimeMean* attribute and a arrival time *Distribution*, and the Server component a *serviceTimeMean* attribute, a service time *Distribution*, a *numServiceUnits* attribute, and a service *Policy*. *has_IATime*, *has_SrvTime*, and *has_ArrivalTimeDistr*,

has_ServiceTimeDistr are subproperties of *has_Attribute* and *has_Distribution* respectively. Every newly added base component in the repository must have a corresponding instance in the COSMO ontology provided by the component developer. Furthermore, new component attributes must be related to previously existing attributes. For example, a new component could have a different attribute name for the "hasIATime" property, e.g "hasInterArrivalT", which should be defined. The COSMO Ontology is written in OWL DL and is developed using Protégé.

B. Objective-based Semantic Validation of Emergent Properties

Our approach consists of two main steps: *identification* of an *emergence set*, and *validation* of the composed model states, as shown in Figure 2. In contrast to current approaches, we focus

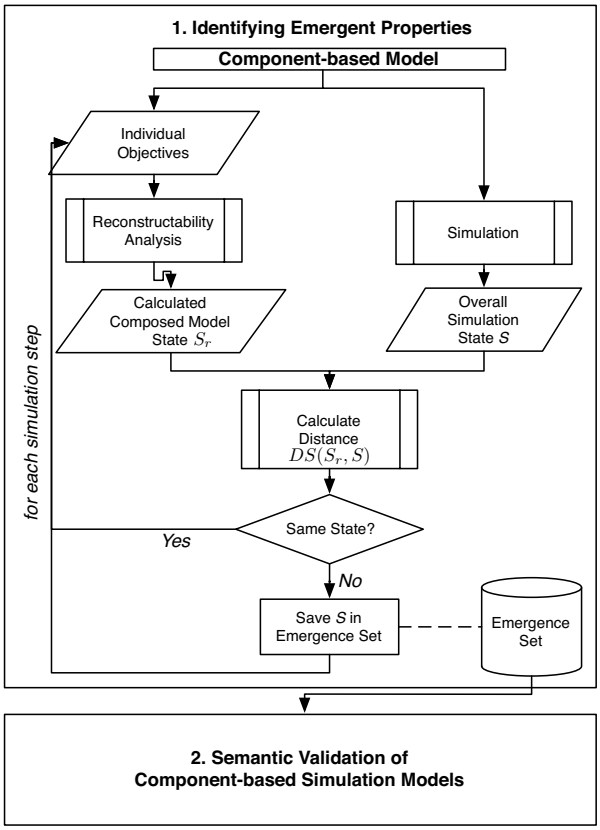


Fig. 2. Objective-based Approach for Identification of Emergent Properties

on the representation of systems in terms of their objectives to facilitate the automated identification and validation of emergence. Firstly, a composition is defined as a "sum" of its constituents and secondly we define objectives to identify emergence. We devise a method of identifying the general properties of the composed model based on the individual properties of the components, by proposing a variation of reconstructability analysis [2] for component-based model development. While reconstructability analysis is designed for abstraction of low-level connection of variables, we propose a higher level of abstract that focuses on component properties and their relations.

1) Identifying Emergent Properties: As shown in Figure 2, a simulation state is a candidate for emergence if it is significantly different from a calculated composed model state. At each simulation step, we propose to calculate the composed model state from the states of its model components. This is done using *reconstruction*, a sub-problem of reconstructability analysis [2], which reconstruct a complex system from variables defining its sub-systems [27]. Once the composed model state S_r is calculated, we compute the semantic difference, $DS(S_r, S)$, between S_r and the simulation state S. If this difference is significant, then simulation state S is added to an emergence set. All components undergo transitions and the process restarts for the next simulation step.

In the reconstruction problem, each model component state $s(C_i)|_o$, is defined as the reunion of all variables *var* that are defined in its objective $o(C_i)$. $s(C_i)|_o$ denotes the model component state is restricted to include only the model component objectives. In contrast, the simulation state S is defined as the reunion of model component states that contain *all* model component attributes and their values (including objectives). This restriction reduces state space explosion in reconstructability analysis. The problem is then to calculate a reconstructed composed model state, S_r, from the states of its model components, $s(C_i)|_o$. Towards this, we propose to use a simplified version of a greedy algorithm adapted from [15]. At each simulation step k, the algorithm calculates

1. `initialize` $S_{rk} = \emptyset$
2. `do`
3. `  select` $\beta \in \{s(C_i)|_o\}$ `to add to` S_{rk}
4. `  when` $\gamma(\beta, S_{rk})$ `is relevant`
5. `until desired size is reached`

Fig. 3. Reconstruction of the Composed Model State S_r

the composed model state S_{rk}. The algorithm starts with an empty set S_{rk}, and continuously adds states from the set $\{s(C_i)|_o | i = 1 \ldots n\}$. The states β that are added to S_{rk} are states that are considered to be relevant by the measure $\gamma(\beta, S_{rk})$, where γ is informally calculated as how statistically different is S_{rk} from $S_{rk} \cup \{\beta\}$, and from previous sets $S_{r_j}, j < k$, calculated at previous simulation steps. Currently, we propose a simple measure of this statistical difference, by looking at the Mann-Whitney-Wilcoxon significance tests over the sets of objective variable values from S_{rk}, $S_{rk} \cup \{\beta\}$, and the sets $S_{r_j}, j < k$. The study of probabilistic models as discussed in [27] is part of our future work.

Next, we compare this theoretic, calculated state S_r, with the simulation state S, defined as the reunion of all model component attributes and their values. We employ a semantic state distance metric, *DS* that looks at the difference between similar attributes as defined in the COSMO ontology. The semantic state distance, *DS*, measures the semantic differences between component attribute values.

Definition 1 (Semantic State Distance)**.** *Let* $s(p) = [state(C_1), \ldots, state(C_n)]$, $s^*(q) = [state(C_1^*), \ldots, state(C_n^*)]$. *The semantic state distance between vectors p and q is defined as*

$$DS(s(p), s^*(q)) = \frac{\sum_{i=1}^{n} |ds(state(C_i), state(C_i^*))|}{n}$$

where $ds(state(C_i), state(C_i^*)) = \frac{\sum_{a_i \in A(C_i), a_j^* \in A(C_j^*)} d(a_i, a_j^*)}{m}$, $A(C_i)$ *is the set of attributes for component* C_i, $m = |A(C_i)|$ *and* $d(a_i, a_j^*)$ *is defined as*

$$d(a_i, a_j^*) = \begin{cases} 0 & \text{if related}(a_i, a_j^*) \text{ and value}(a_i) = \text{value}(a_j^*) \\ 0.5 & \text{if related}(a_i, a_j^*) \text{ and value}(a_i) \neq \text{value}(a_j^*) \\ 1 & \text{if } \nexists a_j^* \in A(C_i^*) \text{ s.t. related}(a_i, a_j^*) = \text{true} \end{cases}$$

where $related(a_i, a_j)$ *signifies that* a_i *and* a_j *are related in the COSMO ontology.*

Our ontology facilitates the calculation of the semantic distance *DS* between composition states. This is done by determining the similarity between component states (*DS*) by calculating the semantic closeness in the ontology of all component attributes (*d*). We propose a semantic state distance rather than a pairwise comparison of values because the state *S* contains a larger number of attributes than the calculate state S_{rk}, which only contains the attributes defining the component objectives. If there is an unacceptable deviation in the observed parameters, i.e. $DS(S_r, S) \geq \varepsilon$, we highlight this state as a possible emergence state and add it to an *emergence set*, *ES* as $ES \cup \{S\}$. We repeat this for the entire simulation run. At the end of the process, the emergence set is shown to the user.

2) Semantic Validation of Emergence: After identifying the emergence set, the component-based model is semantically validated. We propose to validate the composed model using our deny-validity approach [24]. Towards the semantic validation of composed simulation models, our deny-validity approach subjects the composition to a battery of tests that either discard a composed model as invalid, or increase the credibility of the model that is not eliminated [24]. We first eliminate models that have invalid model properties through a process that uses support from model checking and ontologies. At this stage, the validation process focuses on discarding composed models in a three-step approach. Firstly, the component interoperability with respect to exchanged data is validated, using semantically-sugared attribute values from our proposed component-based ontology. Secondly, we employ model checking to validate all possible interleaved execution schedules. From a practical perspective, we consider timeless transitions. Thirdly, we introduce time and validate a meta-simulation of the composed model, using properties specified by the model composer. However, models that pass the first validation stage may have valid properties but may still be invalid when compared with a reference model.

In our previous work, we have proposed a time-based formalism for the representation of the composed model supports the semantic comparison between a composed model and a reference model [24]. In the formalism, a model component is represented as a mathematical function of time and states. We introduce formal definitions of validity that consider closeness with respect to a reference model. We propose a semantic metric relation to evaluate this closeness, considering attribute and state relations in our proposed component-based ontology. To validate emergence, we extend the proposed formalism to analyze states from the emergence set. Previously, our formalism permitted the definition of the simulation execution as a sequence of simulation states, which were represented as a Labeled Transition System (LTS) [23]. The formalism relies on the definition of each model component as a mathematical function, f, according to the definition below.

Definition 2 (Formal Component Representation)**.** *The formal representation for a component* C_i *is a function* $f_i : X_i \rightarrow Y_i$, *where* $X_i = I_i \times S_i \times T_i$, *and* $Y_i = O_i \times S_i \times T_i$. I_i *and* O_i *are the set of input/output messages,* S_i *is the set of states and* T_i *is the set of simulation time intervals at which the component changes state.*

By representing a simulation component as a mathematical function we leveraged on Petty and Weisel's formal theory of composability [19]. However, our approach differs by including *time* and *state* as domain coordinates. Although a three coordinate representation adds to the complexity of the validation process, it allows for a meaningful and detailed definition of a valid model. An example of an LTS representing a simulation run for a simple queueing networks model is given in Figure 4. Each node in the LTS represents an annotated composition state given by the tuple $S_{j=1,m} = [\{state(C_i)_{i=\overline{1,n}}\}, f_{in}, f_{out}]$, where $state(C_i)$ is the state of component C_i, n is the number of components, m is the number of simulation states, f_{in} is the function called to enter this node, and f_{out} is the function called for exiting this node. Edges are the function call f_i in the simulation run, and labels are the tuple <*function_name, duration, output*>, where *duration* represents the function execution time. In this form of labelling, we consider the *duration* rather than the *time* moment when the function begins to execute because the time moments at which functions start to execute are already ordered through the directed nature of the LTS. For example, the LTS in Figure 4 specifies that a transition from the composition state s_1 (defined as a reunion of all model component states) to the composition state s_2 occurs because component C_1, a Source component, executes for a duration of 6 units and outputs the data O_1.

We analyze the states from the emergence set (ES) to understand the sequence of events leading to them. Following the pseudo-code in Figure 5, we identify s as the LTS states that are also in ES. For all of these states, we determine the neighboring states s_{near}, i.e. the states that have either an outgoing edge towards s, or an incoming edge from s. We consider only the neighboring states under the assumption that it is these states that have the greatest impact on the current state. For these states, we calculate the semantic state

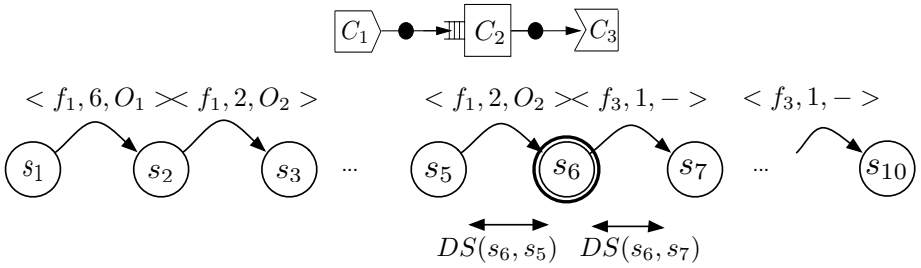


Fig. 4. Validation of States in the Emergent Set

```
1. for each state s in LTS
2.     if s ∈ ES
3.         for each state s_near ∈ neigbours(s)
4.             calculate DS(s, s_near)
5.             if DS(s, s_near) ≥ ε then
6.             Emergent property found
```

Fig. 5. Pseudo-code for the Validation of Emergent Properties

distance *DS*, as defined above. If *DS* has a value higher than a threshold, then the composed model has emergent properties. It is important to highlight here that while for simple examples as the one in Figure 4, the ordering of the states in the LTS implies that each state has one incoming and one outgoing edge, most composed models will have states with more than one incoming, and/or outgoing edges. Moreover, while in the previous step we calculated the distance *DS* between a calculated system state and an observed system state, in this step we calculate the distance *DS* between observed system states that are candidate for emergence, and the rest.

3) Example: Consider for simplicity a model of a flock of birds. Each component abstracts a moving bird, which changes its position based on a set of simple rules that defines its current position and the position of the other birds in the flock. These rules are (i) *separation* - individual bird steer to avoid crowding the other birds in the flock (ii) *alignment* - individual steer towards the average herding of local flockmates and (iii) *cohesion* - individual moves towards the average position of local flockmates. The boid model has been shown to exhibit emergent behavior of flocking, and of flocking after encountering an obstacle, when the flock splits and reunites.

We define the model component objective as:

$$o(b_i) = (description, var(o(b_i)))$$

where

$$description = \textit{Fly northbound with an average speed of}$$

$$\textit{20km/hour};$$

$$var(o(b_i)) = \{direction, speed\}$$

For simplicity and for alignment with the previous example, we discuss a simplified case where a) the speed of each bird is calculated based on the distance from the center of the drawing panel; and b) the calculated state S_{rk} is calculated at each simulation step by considering the reunion of bird states when each bird in is executed in *isolation*. The latter assumption limits the study of systems where individuals can also function independently, as is the case for well-known examples of emergence such as flocks of birds.

The simulation of the flock of birds is executed, and data is collected. Our example employs ten boids that go through a sequence of 20 position changes according to the rules specified above. Initially birds are assigned random positions on the drawing panel. At each simulation step, each birds changes position according to the rules specified above. At each step we collect the position of each individual boid, as well as that of the center of mass of the flock of birds. Since the positions of the entities are given as two coordinate values, we reduce the number of variables by computing the distance d_i from the center of the environment.

If each individual is executed independently, the direction of flight does not change and remains northbound (e.g. "N" in Table I(a)). However, the value of the distance increases rapidly and each individual bird leaves the panel within four simulation steps (e.g. "-" in Table I (a)). Tables I(a) and I(b) present the collated results for individual runs and when the whole group is considered respectively, for the first five birds.

(a) Independent Individuals

Bird 1	**Bird 2**	**Bird 3**	**Bird 4**	**Bird 5**
120, N	114, N	135,N	111, N	98, N
155, N	193, N	146, N	180, N	150, N
360, N	394, N	387, N	380, N	350, N
539, N	571, N	560, N	558, N	538, N
-, N	-, N	-, N	-, N	-, N
-, N	-, N	-, N	-, N	-, N

(b) Entire Group

Bird 1	**Bird 2**	**Bird 3**	**Bird 4**	**Bird 5**
53, S	209, N	128,N	67, S	43, N
45, S	174, NW	112, N	65, S	44, N
40, S	129, NW	88, N	60, SW	43, N
39, S	76, NW	59, N	51, SW	38, N
38, N	27, NW	42, N	34, NW	37, N
54, N	32, NW	48, N	11, NW	57, N
59, N	71, NW	69, NW	13, NW	75, N
56, NW	113, NW	100, NW	42, NW	94, N
58, NW	154, NW	136, NW	79, NW	118, NW
82, NW	195, NW	175, NW	125, NW	151, NW

TABLE I
OBSERVATION OF THE FLOCK OF BIRDS

Any state representing the data in Table (a) will be dif-

ferent from the states in Table (b), with the most significant differences appearing when flocking occurs. This is because without flocking, there are still individuals that follow their own individual flight path, similar to the case when they are considered independently.

IV. Conclusion

Current approaches for the validation of emergent properties can be classified as a-priori such as variable-based and event-based approaches, and a-posteriori such as grammar-based approaches. A-priori approaches assume that we know the emergence properties before-hand. In this paper, we proposed a new a-posteriori, objective-based method that allow us to identify and validate new emergent properties. Different from current approaches, our method specifies a model component in terms of *what* it achieves, rather than *how* it is achieved. The approach is divided into two steps and determines the differences between a calculated system state and an observed system state, through simulation. In the first step, we use the objective-based definition to identify an *emergence set*, which contains observed simulation states for which there are significant differences between a system state calculated through reconstructability analysis, and an observed simulation state. The second step determines the distance between the states in the emergence set and the other states in the simulation run, and states with significant state distance are highlighted as potential emergent. Unlike the grammar-based approach, our objective-based method allows the potential emergent set to be subjected to more rigorous time-based formal mathematical validation. Secondly, we exploited a modified version of reconstructability analysis to address the common state explosion problem in representing component behavior as finite-state machines. Moreover, the flexibility of having elastic and high-level abstraction in the modeling of components increases the potential to identify new emergent properties. This is believed to be more closely associated with the macro-level abstraction of the system under study. Lastly, we demonstrated the application of our approach by extending the component-based simulation model development framework that we previously developed.

Despite our efforts as discussed above, there are a number of open issues that we are currently addressing. Firstly, reconstructability analysis requires several observations of the composed model in order to determine the relevant variables for reconstruction. Secondly, determining the threshold value to distinguish emergence from the set of potential emergence is a challenge. Towards these, we are working towards applying our approach to study road traffic and social networks models among others.

Acknowledgments

This work is supported by the National University of Singapore under grant number R-252-000-470-112.

References

[1] M. Bedau. Weak Emergence. *Philosophical Perspectives*, 11:375399, 1997.

[2] R. Cavallo and G. Klir. Reconstructability Analysis of Multi-dimensional Relations: A Theoretical Basis for Computer-aided Determination of Acceptable System Models. *Int. Journal of General Systems*, 5:143171, 1979.

[3] W. Chan, Y. S. Son, and C. M. Macal. Simulation of Emergent Behavior and Differences Between Agent-Based Simulation and Discrete-Event Simulation. In *Proceedings of the Winter Simulation Conference*, pages 135–150, 2010.

[4] W. K. V. Chan. Interaction Metric of Emergent Behaviors in Agent-based Simulation. In *Proceedings of the Winter Simulation Conference*, pages 357–368, Phoenix, USA, 2011.

[5] C. Chen, S. B. Nagl, and C. D. Clack. Specifying, Detecting and Analysing Emergent Behaviours in Multi-Level Agent-Based Simulations. In *Proceedings of the Summer Computer Simulation Conference*, 2007.

[6] L. Chi. Transplating Social Capital to the Online World: Insights from Two Experimental Studies. *Journal of Organizational Computing and Electronic Commerce*, 19:214–236, 2009.

[7] P. Davis. New Paradigms and Challenges. In *Proceedings of the Winter Simulation Conference*, Orlando , USA, 2005.

[8] U. Fayyad and R. Uthurusamy. Evolving Data Into Mining Solutions for Insights. *Communications of the ACM*, 45, 2002.

[9] S. Floyd and V. Jacobson. The synchronization of Periodic Routing Messages. In *Proceedings of SIGCOMM*, pages 33–44, 1993.

[10] M. Gardner. Mathematical Games, 1970.

[11] R. Gore and P. Reynolds. An Exploration-based Taxonomy for Emergent Behavior Analysis in Simulation. In *Proceedings of the Winter Simulation Conference*, pages 1232–1240, Miami, USA, 2007.

[12] R. Gore and P. Reynolds. Applying Causal Inference to Understand Emergent Behavior. In *Proceedings of the Winter Simulation Conference*, pages 712–721, Miami, USA, 2008.

[13] J. Holland. *Emergence, From Chaos to Order*. Basic Books, 1999.

[14] C. W. Johnson. What are Emergent Properties and How Do They Affect the Engineering of Complex Systems? *Reliability Engineering and System Safety*, 12:1475–1481, 2006.

[15] B. Jones. A Greedy Algorithm for a Generalization of the Reconstruction Problem. *Int. Journal of General Systems*, 11:63–68, 1985.

[16] A. Kubik. Towards a Formalization of Emergence. *Journal of Artificial Life*, 9:41–65, 2003.

[17] J. C. Mogul. Emergent (mis)behavior vs. Complex Software Systems. In *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems*, pages 293–304, New York, USA, 2006.

[18] E. Page and J. Opper. Observations on the Complexity of Composable Simulations. In *Proceedings of the Winter Simulation Conference*, volume 1, pages 553–560, Phoenix, USA, 1999.

[19] M. Petty and E. W. Weisel. A Composability Lexicon. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 181–187, Orlando, USA, 2003.

[20] K. K. Ramakrishnan and H. Yang. The Ethernet Capture Effect: Analysis and Solution. In *Proceedings of the IEEE Local Computer Networks Conference*, Minneapolis, USA, 1994.

[21] C. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. In *Proceedings of ACM SIGGRAPH*, pages 25–34, 1987.

[22] A. K. Seth. Measuring Emergence via Nonlinear Granger Causality. In *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 545–553, 2008.

[23] J. Srba. On the Power of Labels in Transition Systems. In *Proceedings of the 12th International Conference on Concurrency Theory*, pages 277–291, Aalborg, Denmark, 2001.

[24] C. Szabo and Y. Teo. An Approach for Validation of Semantic Composability in Simulation Models. In *Proceedings of the 23rd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation*, pages 3–10, New York, USA, 2009.

[25] C. Szabo and Y. Teo. 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*, page to appear, 2012.

[26] Y. Teo and C. Szabo. CODES: An Integrated Approach to Composable Modeling and Simulation. In *Proceedings of the 41st Annual Simulation Symposium*, pages 103–110, Ottawa, Canada, 2008.

[27] K. Willet and M. Zwick. A Software Architecture for Reconstructability Analysis. *Kybernetes*, 33:97–1008, 2004.