A Time-based Formalism for the Validation of Semantic Composability*

C. Szabo and Y.M. Teo

Department of Computer Science
National University of Singapore
Computing 1, 13 Computing Drive
Singapore 117417

[claudias,teoym]@comp.nus.edu.sg

Simon See

Sun Microsystems Inc.
1 Magazine Road
Singapore 059567, SINGAPORE

simon.see@sun.com

Abstract

Simulation components are semantically composable if the newly composed model is meaningful in terms of expressed behaviors, and achieves the desired objective. The validation of semantic composability is challenging because reused simulation components are heterogeneous in nature and validation must consider various aspects including logical, temporal, and formal. In this paper, we propose a new time-based formal approach for semantic composability validation. Our validation process provides a formal composition validation guarantee by establishing the behavioral equivalence between the composed model and a perfect model. Next, composition behaviors are compared through time using semantically related composition states. We evaluate our formal approach using time complexity and experimental analysis using the CADP analyzer.

1. Introduction

Composability, an appealing approach of growing interest to the simulation community ([11, 17]), aims to reduce the time and cost of developing complex simulations. Simulation composability [15] can be defined as "the capability to select and assemble simulation components in various combinations to satisfy user requirements". Component-based frameworks that employ reused simulation component promise shorter development time and increased flexibility in meeting diverse user needs [15]. However, there are still many challenges in composable simulations, and the most important include mechanisms for the selection and assembly of simulation components, meaningful run-time interoperation between simulation models [26], as well as semantic composability ([2, 11, 15]), and semantic composition validation [3].

*A version of this paper is published in the Proceedings of the 2009 Winter Simulation Conference, IEEE Computer Society Press, Austin, USA, pp. 1411-1422, Dec. 2009. This paper won the ACM SIGSIM Best PhD Student Paper Award, 2009.

Several challenges make the validation of semantic composability an important focus of research in the simulation community in recent years ([15, 23]). Firstly, composition is not a closed operation with respect to validation because valid components do not necessary form valid compositions [1]. Next, reused components are developed for different purposes and when composed may result in emergent properties [7]. Similarly, the context in which a reused component was developed and validated might differ from the new context of the composed model ([2]). Another challenge arises from the various aspects of component interactions that need to be validated. The validation process must address *logical* aspects such as deadlock, safety, and liveness, *temporal* aspects such as the behavior of components and compositions over time, and *formal* aspects such as the need to provide a formal measure of the validity of compositions, also called "figure of merit" [11]. A formal approach to validation is desirable because it offers the possibility of ranking models based on their validity and, when automated in a computer program, eliminates the need of system experts to validate the execution of the composed models [11]. Furthermore, a formal validation process offers the guarantee of provable simulation validity for the benefit of the simulation model users [15]. The main validation techniques of composable simulations include the DEVS formalism [27], Petty and Weisel's formal theory of composability [15], and component abstractions such as BOM [12].

In this paper, we present a formal approach to the validation of semantic composability. Based on our proposed time-based formalism, we provide a formal composition validation guarantee by determining the behavioral equivalence between the composed model and a perfect model over time. The contributions of this paper include:

1. We propose a time-based formalism for the representation of a simulation component as a function of its states over its execution time.
2. We introduce a novel semantic metric relation, V_ε, for comparing simulation executions of a composed model with a perfect model. Based on well-defined concepts in a component-based ontology, V_ε quantifies state similarities and considers only composition states that are semantically related. Through V_ε, the formal guarantee to validity has higher credibility compared with current measures ([15, 24]) because the comparison is done based on both time and semantics, two important considerations in simulation.
3. To validate our approach, we present a theoretical and experimental analysis of our validation process. We analyze the average time complexity of our approach with respect to the number of components in the composed model. From a practical perspective, we present simple and complex examples and analyze the execution time of our validation process.

This paper is organized as follows. We present our approach to formal validation of semantic composability in Section 2. In Section 3, we validate our approach through theoretical analysis and practical experiments. We compare and contrast our approach with current work in Section 4 and present our concluding remarks in Section 5.

2. Proposed Time-based Formalism

The proposed validation process aims to provide a formal measure of composition validity by comparing the composition with a perfect composition made up of perfect components. We consider that for each type of base component there exists a perfect model in the repository, initially provided by domain experts. The perfect base component models describe what the domain experts consider to be the ideal component behavior. The generic descriptions lack specific attributes (e.g. sampling distributions for time attributes) and are without an implementation. We assume that for each base component type (e.g. *Source* in Queueing Networks Application domain) there exist different base component implementations in the repository (e.g. *SourceOpen* - a *Source* component for open queueing network systems). The base component implementations may differ widely from the perfect base component models. The idea of comparing a composed model with a perfect model has been previously explored by [15], by representing components as functions of integer values. However, this representation does not allow complex compositions (e.g. with “fork” connectors). This is because different outputs for the connector branches cannot be specified using a single coordinate functional domain. Moreover the mathematical composition of functions cannot be applied to connector branches. Furthermore, the simulation execution is represented statically based on the component position in the composition. We provide a major improvement by representing components dynamically as functions of states over *time*. Our novel formalism allows for complex models to be validated. Furthermore, simulation execution is represented as a time ordered schedule of component executions. This allows for a more accurate validation process in which the composition execution through time is evaluated.

In this section, we first present our formal definitions of components, simulation, and validity in the context of our formal validation process. Based on our proposed formal definitions, we establish a formal validation process as a sequence of well defined steps. We exemplify our proposed process using an example of a single-server queue.

2.1. Validity

In this section we define components, simulation, and validity in the context of our formal validation process. To facilitate the proposed validation process we separately represent a simulation component using our proposed time-based formalism. The separation of the component specification from the component implementation is widely recognized in the simulation community as an important step towards simulation composability and model reuse ([15, 25]). In our proposed approach, a simulation component is represented as a function of states over time.

Definition 1 (Simulation Component)**.** *The formal representation of a simulation component C_i is a function $f_i : X_i \to Y_i$, 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 leverage on Petty and Weisel’s formal theory of composability [15]. However, our approach greatly differs by including *time* and *state* as domain coordinates. Our three coordinate representation allows for a meaningful and detailed definition of a valid model without affecting the complexity of the validation process. The domain of each functional representation is $X_i = I_i \times S_i \times T_i$. Coordinate I_i represents semantically rich inputs, enriched by our COSMO ontology [22]. Next, S_i represents all possible component states. A component state contains all values of the component attributes. Lastly, T_i represents the set of simulation time moments at which state transitions occur.

Definition 2 (Composition)**.** *Given the components C_i, $i = \overline{1,n}$ formally represented as f_i. The formal representation of the composed model made up of C_i is $M = \{(f_i, f_j) | i \neq j, i, j = \overline{1,n}\}$ with $(f_i, f_j) \in M$ meaning that C_i is connected to C_j in the composed model with C_j requiring input from C_i.*

Definition 3 (Mathematical Composability)**.** *Given a composed model $M = \{(f_i, f_j) | i \neq j, i, j = \overline{1,n}\}$, and the time values when f_i produces output and f_j requires input, $T_i^{out} = \{t_m^{(i)} | 1 \leq m \leq |O_i|\}$, and $T_j^{in} = \{t_n^{(j)} | 1 \leq n \leq |I_j|\}$ respectively. Then f_i and f_j are composable iif there exists the bijective binary relation $R = \{(t_n^{(j)}, t_m^{(i)}) \in T_j^{in} \times T_i^{out} | t_n^{(j)} > t_m^{(j)}\}$.*

Informally, for the component functions to be composable, all sampled time values for components requiring input must be greater than the time moment values for the components that provide them with output. Definition 3 is the usual mathematical composability definition but only considers time moment values from the three coordinate function domain. This is because individual component states are irrelevant at this point in the validation, and input and output data has been previously validated [20].

A simulation represents the execution of the composition over the simulation time.

Definition 4 (Simulation)**.** *The simulation $\mathbb{S}(M)$ of the composed model $M = \{(f_i, f_j) | i \neq j, i, j = \overline{1,n}\}$ is defined formally as the ordered set $\mathbb{S}(M) = \{[\ldots f_i(I_p^i, S_p^i, t_p^i) \to (O_p^i, S_{p+1}^i, t_{p+1}^i), \ldots, f_j(I_q^j, S_q^j, t_q^j) \to (O_q^j, S_{q+1}^j, t_{q+1}^j) \ldots] | t_p^i \leq$*

$t_q^j, t_p^i \leq t_{p+1}^i, t_q^j \leq t_{q+1}^j, i, j \in \overline{1,n},\}$ *where* $I_p^i \in I_i$, $S_p^i, S_{p+1}^i \in S_i$, $t_p^i, t_{p+1}^i \in T_i$, *and* $I_q^j, \in I_j$, $S_q^j, S_{q+1}^j \in S_j$, $t_q^j, t_{q+1}^j \in T_j$.

Informally, the simulation $\mathbb{S}$ of a composition of functions M is defined as the ordered set of the function executions for all components. The set order is based on the time t_p^i at which each function f_i is executed. For example, if $(f_i, f_j) \in M$, then at least one function execution of f_i will come before all function executions of f_j. Thus, the simulation of the composed model is an ordered schedule of the function executions. This provides an accurate representation of the simulation to facilitate the validation process. Previous approaches such as that by [16] employ a *static* simulation description in which components appear in the linear order of aggregation in the composed model. Using our proposed time-based formalism, we obtain a *dynamic* representation of the simulation, in which components appear based on the time moments when they run.

The proposed formal validation approach aims to formally compare between the simulation of the composed model and the simulation of a perfect model. The perfect model is defined below.

Definition 5 (Perfect Model)**.** *Given a composed model of components C_i represented formally as $M = \{(f_i, f_j) | i \neq j, i, j = \overline{1,n}\}$, the perfect model is defined as $M^* = \{(f_i^*, f_j^*) | i \neq j, i, j = \overline{1,n}\}$, where C_i^* formally represented as f_i^* is the corresponding perfect component for component C_i.*

To facilitate the comparison between the composed model simulation, $\mathbb{S}(M)$, and the perfect model simulation, $\mathbb{S}(M^*)$, the two simulations are represented as Labeled Transition Systems (LTS) [18]. Next, we compare the two LTS using the well established theory of bisimulation [14].

Definition 6 (Simulation Representation)**.** *Given a composed model M and its simulation $\mathbb{S}(\mathbb{M})$. The simulation run $\mathbb{S}$ is represented as a Labeled Transition System $L(M) = (N, Act, \rightarrow)$ where N is the set of nodes, $\rightarrow$ is the set of transitions between nodes, and Act is the set of transition labels. In $L(M)$, each node in N represents an annotated composition state given by the tuple $S_{j=\overline{1,r}} = [\{state(C_i)_{i=\overline{1,n}}\}, f_{in}, f_{out}]$, where $state(C_i)$ is the state of component C_i, $r = |\mathbb{S}|$, n is the number of components, f_{in} is the function called to enter this node, and f_{out} is the function called to exit this node. Edges $\rightarrow$ are the function calls f_{in} or f_{out} in the simulation run, and labels $a_i \in Act$ are the tuple* <function_name(f_{out}), duration(f_{out}), output(f_{out})>, *where $duration(f_{out})$ represents the execution time of f_{out}.*

A simulation run is represented as an LTS where nodes represent the entire composition state as a reunion of the individual component states, and edges are labeled to facilitate the validation process. To facilitate accurate comparison between $L(M)$ and the perfect LTS $L(M^*)$, the edge labels contain the name of the function called to exit the node, its duration, and its output. We consider the *duration* rather than the *time* moment when f_{out} begins to execute, because the time moments at which the functions f_{out} start to execute are already ordered through the directed nature of simulation $\mathbb{S}$.

In our proposed formal theory, we consider two possible relations between the simulation of the composed model and the simulation of the perfect model, $L(M)$ and $L(M^*)$ respectively: strong equivalence relation [14] and our proposed semantic parametric metric relation, V_ε. Informally, strong equivalence between $L(M)$ and $L(M^*)$ validates that $L(M)$ is exactly the same or included in $L(M^*)$, including the sequence of the function calls and the edge labels. If this is not possible, we propose the semantic parametric relation V_ε as a weak bisimulation relation. V_ε considers only parts of $L(M)$ and $L(M^*)$ that are semantically close and validates that they appear in the same sequence in $L(M)$ and $L(M^*)$. V_ε is defined below.

Definition 7 (Semantic Parametric Metric Relation)**.** *Let $P \subseteq \{S_1, \ldots, S_n\}$, $Q \subseteq \{S_1^*, \ldots, S_n^*\}$ a subset of the annotated composition states for $L(M)$ and $L(M^*)$ respectively, with $p \in P$, $q \in Q$, $p = [s(p), f_{in}(p), f_{out}(p)]$, $q = [s^*(q), f_{in}^*(q), f_{out}^*(q)]$, with $s(p) = [state(C_1), \ldots, state(C_n)]$ and $s^*(q) = [state(C_1^*), \ldots, state(C_n^*)]$ vectors representing component states. We define the semantic relation with parameter ε, $V_\varepsilon \subseteq P \times Q$, as $V(p,q) = \{(p,q) \in P \times Q | \|p - q\|_\sigma \leq \varepsilon\}$. The semantic vector norm, $\|p - q\|_\sigma$, is defined as*

$$\|p-q\|_\sigma = \frac{DS(s(p), s^*(q)) + \frac{DF(f_{in}(p), f_{in}^*(q)) + DF(f_{out}(p), f_{out}^*(q))}{2}}{2}$$

where $DS(s(p), s^*(q))$ is the semantic distance between composition states, and $DF(f_i, f_j^*)$ is the semantic functional distance between the function names.

The semantic metric relation with parameter ε, V_ε, contains semantically related states between $L(M)$ and $L(M^*)$. Semantically related states are those for which the semantic vector norm, $\|\|_\sigma$, is smaller than the parameter ε. The semantic vector norm has two components, DS and DF. The semantic state distance, DS, measures the semantic differences between component attribute values. The semantic functional distance, DF determines whether the functions that are called to enter and exit the LTS nodes are related.

Definition 8 (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.

Definition 9 (Semantic Function Distance)**.** *Let $f_i(p), f_j^*(q)$ the functions called to enter or exit nodes p and q in $L(M)$ and $L(M^*)$ respectively. The semantic state distance DF is defined as*

$$DF(f_i(p), f_j^*(q)) = \begin{cases} 1, & i \neq j \\ 0, & i = j \end{cases}$$

In the CoDES framework, the COSMO ontology describes component-based simulations within and across simulation domains [22]. The COSMO 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). Informally, V_ε determines whether semantically related states from $L(M)$ and $L(M^*)$ (in terms of composition state - DS, and incoming and outgoing function calls - DF) appear in the same labeled sequence in $L(M)$ and $L(M^*)$ respectively. The above definition is similar to that of Petty and Weisel [15]. However, the fundamental difference and our major improvement comes from forcing the weak bisimulation relation to be V_ε which we previously defined. V_ε is a *semantic* metric relation which considers related composition states according to the COSMO ontology in which a well defined component and attribute hierarchy is present. By representing components as functions of times and states, $L(M)$ and $L(M^*)$ can be compared based on the timed sequences of component executions. Through V_ε, the model can be compared with a perfect model based on rigorously defined concepts in an ontology.

Definition 10 (Validity)**.** *Given the composed model $M = \{(f_i, f_j) | i \neq j, i, j = \overline{1,n}\}$ and its simulation representation $L(M)$, and a perfect model $M^* \{(f_i^*, f_j^*) | i \neq j, i, j = \overline{1,n}\}$ with the simulation representation $L(M^*)$. M is valid iif $(f_i, f_j) \in M$ and $(f_i^*, f_j^*) \in M^*$ are composable respectively (by Definition 3) and there exists a binary relation R between $L(M)$ and $L(M^*)$, with $L(M)\ R\ L(M^*)$ such that R is either a strong equivalence relation [14] or a weak semantic parametric relation, V_ε.*

2.2. Validation Process

Based on the definitions presented above we refine the formal validation process to the five steps presented in Figure 1. The first three steps of the validation process, namely *Unfolding and Sampling*, *Composition*, and *Simulation* are applied separately to the components and perfect components. Components and perfect components annotated with a star symbol (*) from the composition and perfect composition respectively are formally represented as functions of their states over time according to Definition 1. The formal component representations are input to the *Unfolding and Sampling* step where the component representation is adjusted to fit our validation process. Based on the composed model topology, the unfolded representations obtained from the *Unfolding and Sampling* step are composed according to Definition 3 in the *Composition* step. The *Simulation* step applied to the composition and perfect composition results in a composition simulation, $L(M)$, and perfect composition simulation, $L(M^*)$, respectively according to Definition 6. The *Composition* step formally composes the functional representations based on our mathematical composability definition which considers the time moments at which the functions are activated. As such, $L(M)$ and $L(M^*)$ consist of time-ordered simulation schedules of the function executions. Lastly, in the *Validation* step, we first attempt to determine whether $L(M)$ and $L(M^*)$ are exact matches. This is done by determining strong equivalence between $L(M)$ and $L(M^*)$. If strong equivalence is not possible, we introduce the semantic relation V_ε to determine weak equivalence only between related states, i.e. the parts in the two executions that are semantically related. If V_ε is not a weak bisimulation relation between $L(M)$ and $L(M^*)$, then the model is invalid.

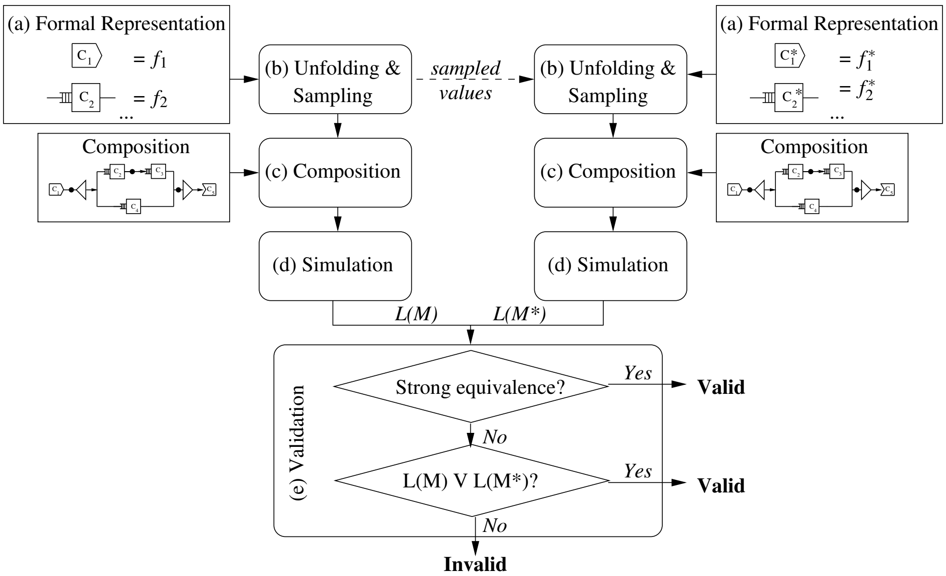

Figure 1. Formal Validation Process

Assumptions For the formal validation process to be realized, a series of assumptions must be in place. Firstly, component information including state machine description must be available in the form of a meta-component in which component attributes and state machine are defined in a standardized format. The component state machine is provided by the component creator when the component is added to the component repository. For example, components in our CoDES framework are represented as a meta-component in a standardized COML format [22]. Next, logical composition properties such as safety and liveness [13] over time have been previously validated. This is the case in the layered validation process in the CoDES framework in which logical properties are first validated in the context of instantaneous transitions and secondly, over time [20]. Lastly, to facilitate the calculation of V_ε, a component-based ontology in which component-based simulation concepts as well as application domain notions are rigorously defined. In the CoDES framework, the COSMO ontology describes component-oriented simulation within and across application domains [22].

For illustration, we apply the above steps on a single-server queue example consisting of Queueing Networks base components as shown in Figure 2. For simplicity, we discuss the first three steps only for the components f_i. For this example, we consider the behavior of the perfect components represented by f_i^* to be the same with respect to input/output transformations to the behavior represented by f_i. The base components C_i might differ from the perfect components C_i^* through additional logging attributes. We formally represent a component as a function of states over time according to Definition 1. The *Unfolding*

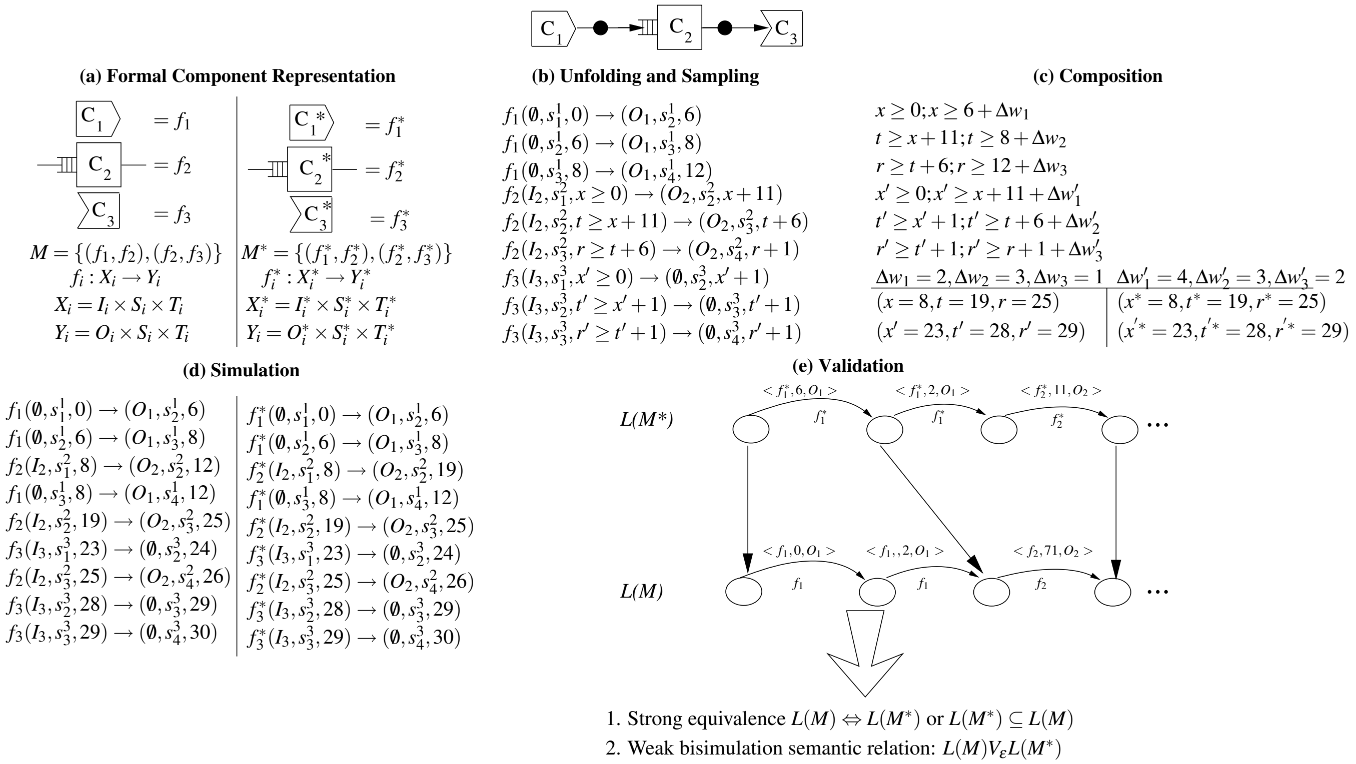

Figure 2. Formal Validation of a Single-Server Queue

and Sampling step in Figure 2(b) unfolds the function definition over a period of simulation time using sampled values. For example, the state machine for meta-component C_1 is defined as $S_1(\Delta t) \longrightarrow O_1 S_1[A_1]$ where Δt is sampled from an exponential distribution with a mean of 4, we have $f_1 : \emptyset \times S_1 \times T_1 \to \{O_1\} \times S_1 \times T_1, f_1(\emptyset, s_i, t) \to (O_1, s'_i, t + \Delta t)$. The above expression is not useful for the *Unfolding and Sampling* step in our approach because during a simulation run t and Δt have specific values. Thus the function call graph is unfolded for an unfolding coefficient of $\tau = 3$ times and sample the values for Δt, using mean values provided by the user. For component C_1 assume that the inter-arrival time is sampled from an exponential distribution with a $mean = 3$. With sampling and an unfolding coefficient of $\tau = 3$ we have $\Delta t = 6, \Delta t = 2, \Delta t = 4$, and thus we can write the sequence: $f_1(\emptyset, s_1^1, 0) \to (O_1, s_2^1, 6), f_1(\emptyset, s_2^1, 6) \to (O_1, s_3^1, 8), f_1(\emptyset, s_3^1, 8) \to (O_1, s_4^1, 12)$, where $f_1(\emptyset, s_1^1, 0) \to (O_1, s_2^1, 6)$ reads as "f_1 *in state* s_1^1 *with no input at time 0, produces output* O_1 *at time 6, while changing its state to* s_2^1". For component C_2, described formally by f_2 assume the service time has an exponential distribution with a mean of $mean = 6$ sampled as $11, 6, 1$. Lastly, we assume component C_3 formalized in f_3 takes 1 unit of time to service each job, so $\Delta t = 1$ for all samples. The values of f_1, f_2, and f_3 are presented in Figure 2(b).

The *Composition* step in Figure 2(c) validates that the functions are mathematically composable according to Definition 2. We obtain constraints on the time variables (x, t, r) and (x', t', r'), considering that f_2 requires input from f_1 before it can proceed, and respectively that f_3 requires input from f_2 before executing. For components that require input to proceed, we also consider the average time spent by messages in the connectors, depicted in Figure 2 by Δw. The constraints in Figure 2(b) are solved by a constraint solver such as Choco [4], resulting in the values $(x = 8, t = 19, r = 25)$ and $(x' = 23, t' = 28, r' = 29)$.

In the *Simulation* step in Figure 2(d), an interleaved simulation run is obtained for model M and for perfect model M^*. The interleaved simulation run orders the function calls based on the time values obtained in the *Composition* step. The simulation runs are represented as Labeled Transition Systems (LTS), $L(M)$ and $L(M^*)$, according to Definition 6.

The *Validation* step is divided into two stages. Firstly, we attempt to prove the equivalence or inclusion between the $L(M)$ and $L(M^*)$ using a *strong* bisimilarity equivalence relation [14], in which only the sequence of labels and states is important. If strong equivalence is not possible, we relax the constraints in the second stage by defining a semantic metric relation V with parameter ε. V_ε considers only semantically related LTS nodes for which our defined semantic distance is smaller than ε. Next, if V_ε is a weak bisimulation metric relation [14] between $L(M)$ and $L(M^*)$, then $C_1, \ldots, C_n$ are semantically composable and $L(M)$ is semantically valid. In the this step, *strong* equivalence between $L(M)$ and $L(M^*)$ is validated using the BISIMULATOR equivalence checker, part of the CADP toolset [6]. For the simple example in Figure 2, the BISIMULATOR returns `true` and as such V_ε need not be calculated.

3. Analysis

3.1. Theoretical Analysis

In this section we analyze the time complexity of our proposed formal validation. Let n be the total number of components in the composed model M. The complexity of our formal validation process, $O_{validation}$, is divided into three main parts:

$$O_{validation} = O_{transform} + O_{compose} + O_{bisimulate} \tag{1}$$

where $O_{transform}$ is the time complexity for the formal component representation, unfolding and sampling, and simulation steps (Step 1 and 3); $O_{compose}$ is the time complexity for the Composition step (Step 2) and $O_{bisimulate}$ is the time complexity for the Validation step (Step 4). The time complexity for the formal component representation and the unfolding and sampling steps, as well as the simulation step is in the worst case $O(n)$. Thus,

$$O_{transform} = O(n) \tag{2}$$

The time complexity of the Composition step is reduced to the time complexity required by a constraint solver implementation to solve the proposed constraints. The constraint satisfaction problem is NP-Complete. However, the algorithm that solves the particular set of constraints from the Composition step has the time complexity of $O(n)$. This is because we require a single solution and it can be obtained by fixing the values for the time moments for the source components (e.g. x in Figure 2) and propagating the values to the rest of the variables. This is permitted by the peculiar nature of the simulation LTS, which, by Definitions 4 and 6 will always have a single edge exiting any state. Therefore,

$$O_{compose} = O(n) \tag{3}$$

For two LTS with N nodes and M transitions, strong and weak bisimilarity between two states can be determined in $O(MN)$ [10]. As such, strong and weak bisimilarity between two LTS can be determined in $O(N^2M)$. For the two LTS that are obtained in the Simulation step, we have $N = \tau n$ and $M = N - 1 = \tau n - 1$, where τ is the unfolding coefficient employed in the *Unfolding and Sampling* step. Thus,

$$O_{bisimulate} = O(\tau^2 \times n^2 \times (\tau n - 1)) = O(n^3) \tag{4}$$

Combining (2) - (4), (1) becomes:

$$O_{validation} = O(n) + O(n) + O(n^3) = O(n^3) \tag{5}$$

Therefore, the complexity of our proposed formal validation is *polynomial* in the number of components.

3.2. Experimental Analysis

The proposed formal validation process is fully implemented in Java. The time value inequalities in the Composition step are solved using the Choco [4] constraint solver. Strong equivalence relation between $L(M)$ and $L(M^*)$ is validated using the CADP toolset [6]. Lastly, V_ε is validated using our proposed algorithm which employs the Jena Reasoner [9] on our proposed COSMO ontology [22] to determine related states. In the previous section we illustrated our approach using a simple single-server queue example. In this section, we discuss a single-server queue example with two classes of jobs, in which the *Server* component has different service times for each class. Next, to show the scalability of our approach, we discuss execution times for the two presented examples and for the validation of a grid scheduler system with eleven components [19]. The grid system has two virtual organizations (VO) sharing a grid meta-scheduler job queue [5]. Each virtual organization consists of a local job scheduler and different types of computational resources modeled as *Server* components. Space constraints prevent us from showing the example here. However, it is discussed in detail in [21]. For the case where the simulation components are very similar to the perfect components, the validation process validates the grid model. However, if we inject several *Server* components with different service times for different job types, then the validation process finds the model as invalid.

Assume that the model to be validated represents a single-server queue presented in Figure 2. The *Source* component (C_1) generates alternatively two classes of jobs that have different service times when serviced by the *Server* component (C_2). The meta-component information relevant for the formal validation process is presented in Table 1.

	C₁	**C₂**	**C₃**
Attribute	$noJobsGenerated = 0$ interArrivalTime: exponential(3)	$noJobsServiced = 0$ serviceTime1 : exponential(6) serviceTime2 : exponential(3) $busy = false$	$noJobsPrinted = 0$ $\Delta printingTime = 1$
Input	-	I_1, constraints: $origin = Source \vert Server \ldots$ $class = C_1$	I_1, constraints: $origin = Server$
	-	I_2, constraints: $origin = Source \vert Server \ldots$ $class = C_2$	-
Output	O_1, constraints: $destination = Server \ldots$ $class = C_1$	O_1, constraints: $destination = Server \vert Sink$	-
	O_2, constraints: $destination = Server \ldots$ $class = C_2$	-	-
State Machine	$S_1(\Delta interArrivalTime) \xrightarrow{[C_1]} S_1 O_1 [A_1]$ $S_1(\Delta interArrivalTime) \xrightarrow{[C_2]} S_1 O_2 [A_2]$	$I_1 S_1 \rightarrow S_2 [A_1; A_3; A_4]$ $I_2 S_1 \rightarrow S_2 [A_1; A_3; A_5]$ $S_2(\Delta serviceTime1) \xrightarrow{C_1} S_1 O_1 [A_2]$ $S_2(\Delta serviceTime2) \xrightarrow{C_2} S_1 O_1 [A_2]$	$I_1 S_1 \rightarrow S_2$ $S_2(\Delta printingTime) \rightarrow S_1 [A_1]$
	$[A_1] = noJobsGenerated++;$ $[C_1] = noJobsGenerated\%2 == 0;$ $[C_2] = noJobsGenerated\%2 == 1;$	$[A_1] = (busy = true);$ $[A_2] = (busy = false);$ $[A_3] = noJobsServiced++;$ $[A_3] = class = C_1;$ $[A_4] = class = C_2;$ $[C_1] = class == C_1;$ $[C_2] = class == C_2;$	$[A_1] = noJobsPrinted++;$

Table 1. Meta-component Information

Table 2 presents the meta-component information for perfect components C_1^*, C_2^*, and C_3^* which are the same as in the example in Section 2. Notice how the time attributes of the perfect component are generically defined. As specified in the validation process, the generic values will be replaced with the sampled values employed in the *Unfolding and Sampling* step for the components part of the composed model.

Following the *Unfolding and Sampling* step we obtain the formal component representation presented

	$\mathbf{C_1^*}$	$\mathbf{C_2^*}$	$\mathbf{C_3^*}$
Attribute	interArrivalTime: generic	serviceTime : generic $busy = false$	$\Delta printingTime = generic$
Input	-	$I, generic$	$I, generic$
Output	$O, generic$	$O, generic$	-
State Machine	$S_1(\Delta interArrivalTime) \to S_2$ $S_2 \to S_1 O$	$IS_1 \to S_2[A_1]$ $S_2(\Delta serviceTime) \to S_1 O[A_2]$	$IS_1 \to S_2$ $S_2(\Delta printingTime) \to S_1$
	-	$[A_1] = (busy = true);$ $[A_2] = (busy = false);$	-

Table 2. Perfect Component Meta-component Information

in Table 3.

Composed Model				Perfect Model			
	Unfold	Δt	**Formula**		**Unfold**	Δt	**Formula**
f_1	1	6	$f_1(\emptyset, s_1^1, 0) \to (O_1, s_2^1, 6)$	f_1^*	1	6	$f_1^*(\emptyset, s_1^1, 0) \to (O_1, s_2^1, 6)$
	2	2	$f_1(\emptyset, s_2^1, 6) \to (O_2, s_3^1, 8)$		2	2	$f_1^*(\emptyset, s_2^1, 6) \to (O_2, s_3^1, 8)$
	3	4	$f_1(\emptyset, s_3^1, 8) \to (O_1, s_4^1, 12)$		3	4	$f_1^*(\emptyset, s_3^1, 8) \to (O_1, s_4^1, 12)$
f_2	1	11	$f_2(I_1, s_1^2, x \geq 0) \to (O_2, s_2^2, x+11)$	f_2^*	1	11	$f_2^*(I_1, s_1^2, x^* \geq 0) \to (O_2, s_2^2, x^*+11)$
	2	2	$f_2(I_2, s_2^2, t \geq x+11) \to (O_2, s_3^2, t+2)$		2	6	$f_2^*(I_2, s_2^2, t^* \geq x^*+11) \to (O_2, s_3^2, t^*+6)$
	3	1	$f_2(I_1, s_3^2, r \geq t+2) \to (O_2, s_4^2, r+1)$		3	1	$f_2^*(I_1, s_3^2, r^* \geq t^*+6) \to (O_2, s_4^2, r^*+1)$
f_3	1	1	$f_3(I_3, s_1^3, x' \geq 0) \to (\emptyset, s_2^3, x'+1)$	f_3^*	1	1	$f_3^*(I_3, s_1^3, x'^* \geq 0) \to (\emptyset, s_2^3, x'^*+1)$
	2	1	$f_3(I_3, s_2^3, t' \geq x'+1) \to (\emptyset, s_3^3, t'+1)$		2	1	$f_3^*(I_3, s_2^3, t'^* \geq x'^*+1) \to (\emptyset, s_3^3, t'^*+1)$
	3	1	$f_3(I_3, s_3^3, r' \geq t'+1) \to (\emptyset, s_4^3, r'+1)$		3	1	$f_3^*(I_3, s_3^3, r' \geq t'^*+1) \to (\emptyset, s_4^3, r'^*+1)$

Table 3. Formal Component Representation

Next, the function composability is validated in the *Composition* step. Following Definition 3 we obtain constraints for the values of x, t, r and x', t', r' respectively. Assuming that the average times spent in the connector queues are $\Delta w_1 = 2, \Delta w_2 = 3, \Delta w_3 = 1$ and $\Delta w'_1 = 4, \Delta w'_2 = 3, \Delta w'_3 = 2$ for f_2 and f_3 respectively, the most trivial constraints that can be derived are:

$$x \geq 6 + \Delta w_1, t \geq x + 11, t \geq 8 + \Delta w_2, r \geq t + 2, r \geq 12 + \Delta w_3$$

$$x' \geq x + 11 + \Delta w'_1, t' \geq x' + 1,$$

$$t' \geq t + 2 + \Delta w'_2, r' \geq t' + 1, r' \geq r + 1 + \Delta w'_3$$

Next, the constraints are solved using the Choco constraint solver with the solution: $(x = 8, t = 19, r = 21)$ and $(x' = 23, t' = 24, r' = 25)$. For the perfect functions f_i^*, the constraint solver returns the solution for the perfect functions time attributes (x^*, t^*, r^*) and (x'^*, t'^*, r'^*): $(x^* = 8, t^* = 19, r^* = 25)$ and $(x'^* = 23, t'^* = 28, r'^* = 29)$.

Interleaved execution schedules are obtained for both composition and perfect composition, as shown in Figure 3(a) and Figure 3(b) respectively. Each interleaved execution is represented as a Labeled Transition

$f_1(\emptyset, s_1^1, 0) \to (O_1, s_2^1, 6)$	$f_1^*(\emptyset, s_1^1, 0) \to (O_1, s_2^1, 6)$
$f_1(\emptyset, s_2^1, 6) \to (O_2, s_3^1, 8)$	$f_1^*(\emptyset, s_2^1, 6) \to (O_1, s_3^1, 8)$
$f_2(I_1, s_1^2, 8) \to (O_2, s_2^2, 19)$	$f_2^*(I_2, s_1^2, 8) \to (O_2, s_2^2, 19)$
$f_1(\emptyset, s_3^1, 8) \to (O_1, s_4^1, 12)$	$f_1^*(\emptyset, s_3^1, 8) \to (O_1, s_4^1, 12)$
$f_2(I_2, s_2^2, 19) \to (O_2, s_3^2, 22)$	$f_2^*(I_2, s_2^2, 19) \to (O_2, s_3^2, 25)$
$f_2(I_1, s_3^2, 22) \to (O_2, s_4^2, 23)$	$f_3^*(I_3, s_1^3, 23) \to (\emptyset, s_2^3, 24)$
$f_3(I_3, s_1^3, 23) \to (\emptyset, s_2^3, 24)$	$f_2^*(I_2, s_3^2, 25) \to (O_2, s_4^2, 26)$
$f_3(I_3, s_2^3, 24) \to (\emptyset, s_3^3, 25)$	$f_3^*(I_3, s_2^3, 28) \to (\emptyset, s_3^3, 29)$
$f_3(I_3, s_3^3, 25) \to (\emptyset, s_4^3, 26)$	$f_3^*(I_3, s_3^3, 29) \to (\emptyset, s_4^3, 30)$
(a) Composition	(b) Perfect Composition

Figure 3. Interleaved Execution Schedules

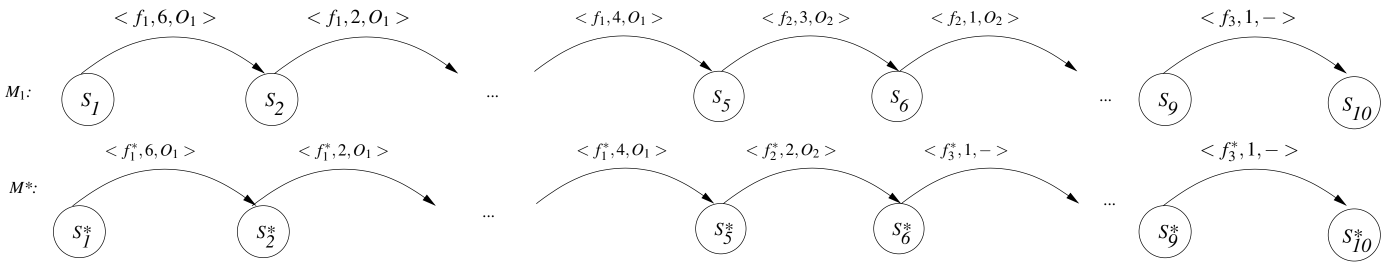

Figure 4. LTS Representation of Model Execution

System, $L(M)$ and $L(M^*)$ respectively, as shown in Figure 4. It is evident that the two LTS are not strongly equivalent (see the outgoing labels from S_6 and S_6^*), hence the BISIMULATOR tool returns `false`. We calculate the semantic metric relation V_ε for $\varepsilon = 0.5$ and obtain the following related nodes: $V_\varepsilon = \{(S_1,S_1^*), (S_2,S_2^*), (S_3,S_3^*), (S_4,S_4^*), (S_3,S_5^*), (S_5,S_5^*), (S_7,S_6^*), (S_7,S_8^*), (S_9,S_9^*), (S_{10},S_9^*), (S_{10},S_{10}^*)\}$, with $\{\|S_i - S_j^*\|_\sigma = 0.41 | \forall(i,j) \neq (5,5)\}$ and $\{\|S_5 - S_5^*\|_\sigma = 0.45\}$. For these values of V_ε we can conclude that the model is not valid since V_ε is not a weak bisimulation relation between $L(M_1)$ and $L(M^*)$.

Table 4 presents the execution times for the examples discussed in this paper, namely a single-server queue in which the components are very similar to the perfect components, a single-server queue with two classes of jobs, a grid system with two virtual organizations in which components are very similar to the perfect components, and lastly a grid system with two classes of jobs. We differentiate between the execution times of the *Unfolding & Sampling, Composition*, and *Simulation* steps (*USCS*), of the *Strong Equivalence Validation* step using the BISIMULATOR tool, and of the V_ε validation where it is necessary. The last column shows the total execution time.

Model	**Number of Components**	**Number of LTS States**	**Result**	**Execution Times (s)**			
				USCS	*Strong Equivalence*	V_ε	**Total**
Single-Server Queue	3	12	`valid`	0.95	1.02	-	1.97
Single-Server Queue with Two Classes of Jobs	3	12	`invalid`	1.04	0.97	2.71	4.72
Grid System	11	51	`valid`	4.58	0.96	-	5.54
Grid System with Two Classes of Jobs	11	51	`invalid`	5.17	1.05	2.07	8.29

Table 4. Validation Process Execution Times

The above examples raise some interesting issues. Firstly, there is the well known difference between what system experts perceive as *valid* and what can be defined in a computer system as a *valid model* for it to validate automatically and independently. In our formal approach, a valid model is one that is *close enough* with respect to the states, sequence and duration of component execution, to a perfect model. Yet, what exactly is close enough (i.e. the values of ε), as with all thresholds, remains an open problem. Furthermore, because of the component-based nature of the validation process, it is difficult to translate the notion of valid/invalid into easier to grasp simulation concepts. Next is the problem of perfect models. While it is acceptable to assume their existence, their origin and content is still an open question. For example, the single-server queue with two classes of jobs described above would be considered valid if the perfect model for the Server component would contain two sampling time intervals instead of a single one. Previous approaches such as Petty and Weisel's [15] do not consider the nature of the perfect components. It is the first time that the exact structure of the perfect models has been studied. Lastly, the impact of a different semantic distance *DS* on the weak semantic bisimulation relation remains to be studied.

4. Related Work

Petty and Weisel pioneered a formal theory of composability validation which allows for a composed simulation model to be checked for semantic validity [15]. A composition is modeled as a mathematical functional composition. The simulation of a composition is represented as an LTS where nodes are model states, edges are function executions, and labels are model inputs. A composition is valid if and only if its simulation is close by a relation to the simulation of a perfect model. In the Petty and Weisel approach, time is not modeled and the function representing a component makes an instantaneous transition from input to output. This permits only a *static* representation of the composition. Furthermore, the LTS representation considers the functions strictly in the order they appear in the mathematical composition, which might not be representative for complex compositions. In contrast, we propose a new formal component definition where states change over *time*. Based on this definition, we represent composition simulations as interleaved schedules of component execution, considering the execution duration and output as labels in the simulation LTS. Thus our approach has the advantage of representing the dynamic change of the entire simulation over time. To provide a more accurate measure of validity, we consider semantically related composition states in the definition of V_ε. This is not be possible in the Petty and Weisel approach where a component is abstracted as a one-dimensional integer domain function.

DEVS (Discrete Event System Specification) [27] is a formalism derived from general system theory and is designed to describe the structure and behavior of a system. In DEVS, a system is modeled as a blackbox with state, input and output ports. For validation, compositions of DEVS models are represented in the Z specification language [24]. A theorem proving tool based on Z such as Z/EVES is used to verify the model and discover hidden properties. Ambiguities, conflicts and inconsistencies can be discovered in the specification. However, the Z specification language lacks time modeling, one of the most important attributes in DEVS models.

A third approach to composition validation [12] uses the Base Object Model (BOM) [8] as a component abstraction. A BOM captures component behavior information including participating entities and their state machines, and information about the possible usage scenarios of the component. This approach assumes that a detailed user specified composition scenario exists to represent a valid composition. The scenario includes the sequence of component execution, as well as events and parameter names for interacting components. Component discovery is done based on the specified scenario. A valid composition of discovered components is one in which the sequence of actions or events is the same as or includes the sequence specified in the scenario. However, the somewhat informal validation process includes the composition and execution of discovered components in *all* possible combinations in order to be compared with the specified scenario, which leads to a costly implementation. Furthermore, a detailed execution scenario might not be available from the model composer.

5. Conclusion

We propose a formal approach for validation of semantic composability. We introduce a novel time-based formalism where a simulation component is represented as a function of its states over time. Based on our formal definitions of composition, simulation, and validity, we refine and specialize the formal validation process to a form applicable to environments for component-based simulation development in which time and state are of paramount importance. In our five-step formal validation process, the behavior of the composed model over time is compared to the behavior of a perfect model. Component functions are unfolded using sampled values. The mathematical composition of the component functions is validated using existing constraint solvers. Simulations of compositions are then represented as interleaved timed execution schedules. The validation process formally compares the composition execution schedule to that of a perfect composition derived from perfect components. The comparison determines the equivalence of the schedules based on a new semantic metric relation, that considers semantically related composition states. This is in contrast to Petty and Weisel's work in which the LTS representation considers function calls in the static order as they appear in the mathematical composition. Furthermore, our time-based formalism permits the representation of complex systems with "fork" and "join" topologies. Our theoretical analysis shows that the validation process has polynomial complexity and our execution time analysis shows that our approach is scalable. We have fully implemented and integrated the validation process in our CoDES component-based framework. This paper addresses the semantic validation of simulation model developed using base components. We are extending the formal validation process to cover the more complex reused model components.

References

[1] O. Balci. Verification, validation and accreditation of simulation models. In S. Henderson, B. Biller, M. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, editors, *Proceedings of the 1997 Winter Simulation Conference*, pages 135–141, Piscataway, New Jersey, 1997. Institute of Electrical and Electronics Engineers, Inc.

[2] R. Bartholet, D. Brogan, P. Reynolds, and J. Carnahan. In search of the philosopher's stone: simulation composability versus component-based software design. In *Fall Simulation Interoperability Workshop*, Orlando, USA, 2004.

[3] P. Davis and A. Tolk. Observations on new developments in composability and multi-resolution mModelling. In S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, editors, *Proceedings of the 2007 Winter Simulation Conference*, pages 859–870, Piscataway, New Jersey, 2007. Institute of Electrical and Electronics Engineers, Inc.

[4] F. Laburthe and N. Jussien. Choco constraint system, 2009. http://sourceforge.net/projects/choco/.

[5] I. Foster, C. Kesselman, and S. Tuecke. *Grid computing, making the global infrastructure a reality*. John Wiley and Sons, 2003.

[6] H. Garavel, F. Lang, R. Mateescu, and W. Serwe. CADP 2006: A toolbox for the construction and analysis of distributed processes. In *Proceedings of the 19th International Conference on Computer Aided Verification*, pages 158–163, Berlin, Germany, 2007.

[7] R. Gore and P. Reynolds. Applying causal inference to understand emergent behavior. In S. J. Mason, R. Hill, L. Moench, and O. Rose, editors, *Proceedings of the 2008 Winter Simulation Conference*, pages 712–721, Piscataway, New Jersey, 2008. Institute of Electrical and Electronics Engineers, Inc.

[8] P. Gustavson and L. Root. Object model use cases: a mechanism for capturing requirements and supporting BOM reuse. In *Spring Simulation Interoperability Workshop*, Orlando, USA, 1999.

[9] HP Labs. Jena - a semantic web framework for Java, 2002. http://jena.sourceforge.net.

[10] P. Kanellakis and S. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Compuation*, 86(1):43–68, 1990.

[11] S. Kasputis and H. Ng. Composable simulations. In J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 1577–1584, Piscataway, New Jersey, 2000. Institute of Electrical and Electronics Engineers, Inc.

[12] F. Moradi, R. Ayani, S. Mokarizadeh, G. H. A. Shahmirzadi, and G. Tan. A rule-based approach to syntactic and semantic composition of BOMs. In *Proceedings of the 11th IEEE Symposium on Distributed Simulation and Real-Time Applications*, pages 145–155, Crete, Greece, 2007.

[13] S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):455–495, 1982.

[14] D. Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pages 167–183, Karlsruhe, Germany, 1981.

[15] M. Petty and E. Weisel. A composability lexicon. In *Proceedings of the Spring Simulation Interoperability Workshop*, pages 181–187, Orlando, USA, 2003.

[16] M. Petty and E. Weisel. Basis for a theory of semantic composability. In *Proceedings of the Spring Simulation Interoperability Workshop*, Orlando, USA, 2003.

[17] M. Pidd. Simulation software and model reuse: A polemic. In R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, editors, *Proceedings of the 2004 Winter Simulation Conference*, pages 772–775, Piscataway, New Jersey, 2004. Institute of Electrical and Electronics Engineers, Inc.

[18] 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.

[19] C. Szabo and Y. Teo. On syntactic composability and model reuse. In *Proceedings of the International Conference on Modeling and Simulation*, pages 230–237, Phuket, Thailand, 2007.

[20] 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, Lake Placid, USA, 2009.

[21] C. Szabo and Y. M. Teo. Validation of semantic composability in simulation models. Technical Report APSTC-TR-2009-01, Sun Microsystems, 2009.

[22] 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.

[23] A. Tolk. What comes after the semantic web - PADS implications for the dynamic web. In *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation*, pages 55–62, Singapore, 2006.

[24] M. K. Traore. Analyzing static and temporal properties of simulation models. In L. R. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, editors, *Proceedings of the 2006 Winter Simulation Conference*, pages 897–904, Piscataway, New Jersey, 2006. Institute of Electrical and Electronics Engineers, Inc.

[25] L. Yilmaz. On the need for contextualized introspective simulation models to improve reuse and composability of defense simulations. *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 1(3):141–151, 2004.

[26] L. Yilmaz and S. Paspuleti. Toward a meta-level framework for agent-supported interoperation of defense simulations. *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 2(3):161–175, 2005.

[27] B. Ziegler, H. Prahofer, and T. Kim. *Theory of modeling and simulation*. Academic Press, 2^{nd} edition, 2000.