A Formal Approach to Composability

Mikel D. Petty, Eric W. Weisel, and Roland R. Mielke
Virginia Modeling, Analysis and Simulation Center
Old Dominion University
Suffolk, Virginia
mpetty@vmasc.odu.edu, eweise@odu.edu, rmielke@odu.edu

ABSTRACT

Composability is the capability to select and assemble simulation components in various combinations into simulation systems to satisfy specific user requirements. The defining characteristic of composability is the ability to combine and recombine simulation components into different simulation systems. Two types of composability are considered: syntactic and semantic. Syntactic composability is the actual implementation of composability and requires that the composable components be constructed so that their implementation details are compatible for the different configurations that might be composed. In contrast, semantic composability is a question of whether the models that make up the composed simulation system can be meaningfully composed, that is, if their combined computation is valid. There has been significant work on syntactic composability but almost none on semantic composability. We have developed a formal theory of semantic composability that enables the determination, in a formal way, of whether a composition of models is valid. In this paper we provide an overview of the essentials of the theory. Definitions of model, simulation, validity and composability are stated and arguments that the definitions are appropriate for the purpose are given. The apparent disconnect between formal reasoning and semantics is resolved using the definition of validity. Theorems regarding the conditions under which compositions of models are valid and the computational complexity of determining the validity of compositions are described.

ABOUT THE AUTHORS

Mikel D. Petty is Chief Scientist of the Virginia Modeling, Analysis & Simulation Center at Old Dominion University. He received a Ph.D. in Computer Science from the University of Central Florida. Dr. Petty has worked in modeling and simulation research and development since 1990 in the areas of simulation interoperability, computer generated forces, multi-resolution simulation, and applications of theory to simulation. He previously worked for the Institute for Simulation and Training at the University of Central Florida. He has served as a member of a National Research Council committee on modeling and simulation and is currently an Area Editor of the journal SIMULATION: Transactions of the Society for Modeling and Simulation International.

Eric W. Weisel is a Project Scientist at the Virginia Modeling, Analysis & Simulation Center at Old Dominion University. He received a M.S. in Operations Research from the Florida Institute of Technology in 1995 and a B.S. in Mathematics from the United States Naval Academy in 1988. He is currently a Ph.D. candidate in Modeling and Simulation at Old Dominion University. His research interests include simulation formalism and composability. He is also President of WernerAnderson, Inc., a private veteran-owned small business that performs research and development in the areas of defense-related operations research and simulation. He previously served as a U.S. Navy submarine officer with experience in nuclear engineering, operations management, navigation, and joint operations.

Roland R. Mielke is Professor of Electrical and Computer Engineering at Old Dominion University and holds the designation University Professor. He also serves as Technical Director for the Virginia Modeling, Analysis and Simulation Center. Dr. Mielke earned B.S., M.S., and Ph.D. degrees, all in Electrical Engineering, from the University of Wisconsin – Madison. His research interests are in the areas of systems theory and simulation and include simulation methodologies, system modeling, composability theory, discrete event simulation, and the application of simulation to the development of enterprise decision support tools.
A Formal Approach to Composability

Mikel D. Petty, Eric W. Weisel, and Roland R. Mielke
Virginia Modeling, Analysis and Simulation Center
Old Dominion University
Suffolk, Virginia
mpetty@vmasc.odu.edu, eweisel@odu.edu, rmielke@odu.edu

INTRODUCTION

Composability is the capability to select and assemble simulation components in various combinations into valid simulation systems to satisfy specific user requirements (Petty, 2003a). The defining characteristic of composability is that different simulation systems can be composed at configuration time in a variety of ways, each suited to some distinct purpose, and the different possible compositions will be valid simulation systems. Composability is more than just the ability to put simulations together from components; it is the ability to combine and recombine, to configure and reconfigure, sets of components from those available into different simulation systems to meet different needs. The process of composing different simulation systems from components is illustrated notionally in Figure 1.

There are two types of composability: syntactic and semantic. Syntactic and semantic composability have also been called engineering and modeling composability, respectively (Petty, 2003a). Syntactic (engineering) composability is the actual implementation of composability; it requires that the composable components be constructed so that their implementation details, such as parameter passing mechanisms, external data accesses, and timing assumptions are compatible for all of the different configurations that might be composed. The question in syntactic composability is whether the components can be connected. In contrast, semantic (modeling) composability is a question of whether the models that make up the composed simulation system can be meaningfully composed; that is, if the combined computation is semantically valid. It is possible that two components may be syntactically linked, so that one can pass data to the other, but semantically invalid, if the data produced by the first component are not within the bounds that can be accepted validly by the other.

The purpose of this paper is to provide a conceptual overview of a formal theory of semantic composability. In the second section, formal definitions for several important terms, including model, simulation, and labeled transition system, are presented and illustrated using simple examples. In the third section, the concept of validity is defined formally using labeled transition systems and a mathematical construct called bisimulation. Then, in the fourth section, model composition is presented. The validity of composite models formed from the composition of valid models is investigated. The conclusion is presented in the final section.

DEFINITIONS

Formal definitions of model and simulation, suitable for formal reasoning, are presented in this section. In addition, a less commonly used construct called a labeled transition system is defined and used to demonstrate the relationship between model and simulation. The definitions are compared to informal definitions in general use. Rationale for the new formal definitions is described and illustrative examples are presented.

Model

The official definition of model utilized by the Department of Defense is as follows.

Model. A model is a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon or process (DOD, 1996), (DOD, 1998).

In this definition, a model is described as a representation of a natural system. The identity of the natural system is
known and the form of the model is left open. While intuitively useful, this definition is insufficiently formal to use as the basis for a formal theory. A definition of model that effectively reverses the informal definition’s specificity with respect to purpose and ambiguity with respect to form is proposed. In other words, the proposed definition of model precisely specifies the model form and is ambiguous as to purpose. The notion of a model’s purpose is recaptured later through the definition of validity. The proposed definition is as follows.

**Model.** A model is a computable function.

Because this definition forms the basis for a formal theory, it is crucial to examine it closely. The term function used in the definition refers to the formal mathematical definition of a partial function. The terms function and partial function are defined in the following.

**Function.** A function from set \( X \) into set \( Y \) is a rule \( f \) that assigns to every member \( x \) of set \( X \) a unique member \( y = f(x) \) of set \( Y \). The set \( X \) is called the domain of the function and the set \( Y \) is called the codomain of the function (Hu, 1969).

**Partial Function.** A partial function from set \( X \) into set \( Y \) is defined similarly to a function from set \( X \) into set \( Y \), but the rule \( f \) may not be defined for every element of \( X \) (Hein, 2002).

To avoid confusion, in this paper the word function is used to mean partial function; the term total function is used to indicate a function defined for all elements of the domain.

The term “computable” in the definition of model makes reference to the formal definition from computability theory. Informally, a computable function is a function that can be computed in a finite number of steps by an algorithm, or equivalently, by a Turing machine or a computer. Note that computable functions are a subset of all functions; that is, some functions are not computable as proven by Turing (Turing, 1937), (Davis, 1982). Formal definitions of a computable function are available in the literature (Davis, 1982), (Davis, 1994), (Sommerhalder, 1988); those definitions will not be repeated here, but will be used for the theory.

Defining models as computable functions in no way excludes any type of simulation component that may be of interest; every process that runs on a digital computer is ultimately a computable function. The notion of a computable function generally is assumed both in the informal definition of model and the way the term is used by simulationists. At time \( t \), a model has state \( S \). The model accepts input \( I \) and produces output \( O \). In addition, the model calculates the new model state that is to be used for the next computation occurring at time \( t + dt \), where \( dt \) denotes the model computation time interval. Using \( M \) to denote the function rule, a model is defined by the functional notation \( M : X \rightarrow Y \) where \( X = (S \times I) \) = domain set and \( Y = (S \times O) \) = codomain set. The notation \( S \times I \) denotes the Cartesian product of \( S \) and \( I \) and is the set of all ordered pairs \((s, i)\) such that \( s \in S \) and \( i \in I \). When convenient, \( M \) can be partitioned to form the state model \( M_S : (S \times I) \rightarrow (S) \) and the output model \( M_O : (S \times I) \rightarrow (O) \). Complex models, e.g., JSAF, may appear to be doing more than this, but because they are computer programs, ultimately they are computing functions.

The domain set \( X \) and the codomain set \( Y \) are defined to be vectors of non-negative integers. This definition restricts the inputs and outputs of models to integers, and specifies vectors of integers, instead of single variables, or matrices. Both of these points can be justified; we first consider the restriction to integers. Practically speaking, for a theory of semantic composability, we are ultimately interested in models that can be implemented and run on digital computers. All of the values that can be represented on such computers are constructed from bits (0s and 1s) and so are integers. The so-called “real numbers” available in most programming languages are in fact integer approximations to real numbers. Theoretically, the restriction to integers is consistent with the assumptions of computability theory (Sommerhalder, 1988), (Barrow, 1992).

Specifying the inputs, outputs and states of models to be vectors, instead of single integers or matrices, is easier to justify. Single integers, vectors of integers, and matrices of integers of any dimension are all equivalent. Vectors of integers can be mapped one-to-one to single integers using a suitable variant of Cantor’s method for mapping rational numbers, which can be represented as vectors with two elements, to single integers (Hein, 2002). A similar argument works for matrices of integers to vectors. Indeed, different presentations of computability theory use different choices; computable functions have been defined using both multi-variable functions, essentially functions on vectors (Davis, 1994), and functions on single integers (Sommerhalder, 1988). We select vectors, rather than single integers or matrices, because it is simple to distinguish between input, output and state variables as elements of vectors.

**Simulation**

The official definition of simulation is as follows.

**Simulation.** Simulation is a method for implementing a model over time. Simulation also is a technique for testing, analysis or
training in which real world systems are used, or where a model reproduces real world and conceptual systems (DOD, 1996), (DOD, 1998).

In the official definition of simulation, “implementing” actually seems to mean “executing”. That is the sense of the term as commonly used; a simulation is an execution of a model over simulated time. For the theory developed here, the following definition of simulation is proposed.

**Simulation.** A simulation is a sequence of executions of a model.

Like the proposed definition of model, this definition of simulation is stripped of all explicit mention of the simulation representing anything, such as a real-world system. This has been done deliberately because defining a model or simulation as representing something is assuming validity. Validity is a property that models and simulations might or might not possess, not something that they should be defined or assumed to possess. Of course, it is generally intended that a simulation is an execution of a valid model, but that is not where a formal reasoning process should start.

Simulation as a sequence of model executions is shown in Figure 2. This diagram is adapted from a description of the execution of a synchronous system that conveys the sense of simulation (Borälv, 1999) (Halbwachs, 1991). In order to initiate model execution, an initial state value and a sequence of inputs must be specified. Then, the model M is executed iteratively. At each execution, the model accepts input from the previous step, denoted s for state, and from outside the system, denoted i for input. The model computes the next state of the system that is passed to the next iteration, again denoted s, and output that is made available outside the system, denoted o. The input i, the state s, and the output o are all vectors of integers as previously defined. The computation at step k is specified by \( (s_k, o_k) = M(s_{k-1}, i_{k-1}) \). Input from and output to outside the simulation are included to allow for interactive simulation. This is consistent with both the official definition of simulation and with intuition.

**Labeled Transition System**

A labeled transition system (LTS) (Roggenbach, 2000) is a concept drawn from theoretical computer science that historically has seen little use in the applied simulation community. However, the LTS provides a useful alternate representation of models and is essential to the formal definition of validity presented in the next section. Thus, we define and illustrate the LTS concept in this section.

A labeled transition system is represented by a directed graph with nodes corresponding to the elements of SST and edges, having labels from IIT, corresponding to the elements of \( \rightarrow_T \). For example, suppose \( s_p, s_q \in SST \) and let \( \rightarrow_T e \rightarrow e \rightarrow_T \rightarrow_T (s_p, i_k) \rightarrow_T (s_q) \). Then the directed graph corresponding to these elements is shown in Figure 3.

![Figure 3. Example Graph](image)

It is apparent that there is a one-to-one correspondence between a state model \( M_S : (S \times I) \rightarrow (S) \) and the associated labeled transition system defined by \( L(M_S) = (S, I, M_S) \). In fact, \( L(M_S) \) is just one more way of displaying the function \( M_S \). The nature of this correspondence is illustrated through a simple example. Consider a model of the Mod2 Counter shown in Figure 4.

![Figure 4. Mod2 Counter Model](image)

The state for this system is \( s = (\text{count}) \) where count has value 0 or 1. The signal input has value 0 or 1 and the model is to count the number of 1s present in this input. The control input has value 0 or 1 and is used to either enable counting (1) or disable counting (0). The reset output has value 0 or 1. The output is 1 to indicate the occurrence of a count reset (0 \( \rightarrow \) 1 state transition) and is
0 otherwise. The model M is expressed in tabular form in Table 1; in functional form, M_S and M_O are given by

\[
\text{Next State} = \text{State} \cdot \text{Control} + \text{State} \cdot \text{Signal} + \text{State} \cdot \text{Control} \cdot \text{Signal} \\
\text{Output} = \text{State} \cdot \text{Control} \cdot \text{Signal}
\]

where \( \cdot \) and \( + \) denote the Boolean operations of AND and OR, respectively, and \( (\cdot) \) denotes complement. The corresponding labeled transition system \( L(M_S) \) is shown in Figure 5.

Table 1. Function Definition Table for Mod2 Counter

<table>
<thead>
<tr>
<th>State</th>
<th>Control</th>
<th>Signal</th>
<th>N. State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5. Labeled Transition System for Mod2 Counter

Simulation and Labeled Transition Systems

A simulation, as an execution of a model, can be understood in terms of the LTS representation of that model. Each execution of the model results in a transition from one state to the next state. The sequence of state transitions corresponding to a sequence of model executions is called a trajectory. The graph of a trajectory consists of an alternating sequence of nodes representing states and transitions representing model executions for specific inputs. The trajectory corresponding to the model execution of Figure 2 is shown in Figure 6. Trajectories are important in defining validity because, intuitively, valid models should exhibit similar trajectories for similar initial states and input sequences.

VALIDITY

A model typically is intended to represent some real-world or notional system. We call this system the natural system. For a given set of initial conditions and a set of inputs, a model for the natural system is to exhibit behavior that is close to the behavior of the natural system. Intuitively, the validity of the model is based upon how closely these behaviors match. Thus, a formal, quantitative definition of validity must include a measure of this closeness of behaviors. In this section, we develop such a definition. First, the behavior of the natural system is captured by defining a perfect model. The behavior of the perfect model is compared to the behavior of a candidate model using the concept of weak bisimulation. Two models are weakly bisimilar if, for a given set of initial conditions and an input set, the models exhibit similar trajectories in their respective LTSs when simulated. A measure of closeness is achieved by requiring the bisimulation to have special properties; here we include equivalence relations and metric relations, but others are possible.

First, the concept of a perfect model is introduced.

**Perfect Model.** A perfect model is a notional model whose simulation for some initial state and input sequence exactly matches the execution of the modeled natural system.

The perfect model is described by the notation \( M^*:X^* \rightarrow Y^* \). It should be noted that a complete representation of \( M^* \) is usually not available; rather, \( M^* \) often must be approximated by making observations, either physical or notional, of the natural system. The validity of other models for the natural system is measured with respect to the perfect model.

In order for another model \( M \) to represent \( M^* \), \( M \) should exhibit behavior similar to that of \( M^* \) for some specified set of initial conditions and some specified input set. A formal method for characterizing similar behavior is to compare the labeled transitions systems for \( M \) and \( M^* \). Two models exhibit similar behaviors for an input set if they generate similar trajectories when simulated. The formal approach for comparing all possible trajectories for a given input set is the concept of weak bisimulation. The formal definition of weak bisimulation is as follows.

**Weak Bisimulation.** Let \( P = (S_P, I, \rightarrow_P) \) and \( Q = (S_Q, I, \rightarrow_Q) \) be two labeled transition systems over the common input set \( I \). A
relation $R \subseteq (P \times Q)$ is a weak bisimulation if and only if, for all $(p, q) \in R$ and $i \in (I \cup \{\tau\})$,

1. if $p \xrightarrow{i} p'$, then there exists a $q' \in Q$ such that $q \Rightarrow q'$ and $(p', q') \in R$, and
2. if $q \xrightarrow{i} q'$, then there exists a $p' \in P$ such that $p \Rightarrow p'$ and $(p', q') \in R$,

where $\hat{i}$ denotes the sequence obtained by deleting all occurrences of $\tau$ actions from $I$ and $p \Rightarrow p'$ means $p \rightarrow \cdots \rightarrow p'$ for some $u, v \geq 0$.

A $\tau$ action, also called an internal action, is a state transition that occurs without application of an external input. If a weak bisimulation $R$ exists, then $P$ and $Q$ are said to be weakly bisimilar. This is denoted by $P \approx_Q R$.

Informally, two labeled transition systems $P$ and $Q$ over a common set of inputs $I$ are weakly bisimilar if every transition in $P$ (Q) is matched by an identical transition in $Q$ (P), possibly preceded and/or succeeded by zero or more internal actions. The definition is recursive so that the resultant states also must be weakly bisimilar (Roop, 2001).

For $M$ to be a valid model for $M^*$, not only must the models exhibit similar trajectories, but in addition the model outputs at each execution, $Y$ and $Y^*$, must be sufficiently close. A bisimulation is a relation $R$ between corresponding nodes (model states) in $L(M)$ and $L(M^*)$; by additionally attributing a measure of closeness to $R$, a quantitative measure of validity results. Two relations that can be used to measure closeness of model states, the equivalence relation and the metric relation, are defined next. Then, the formal definition of validity is presented.

**Equivalence Relation.** An equivalence relation $E$ on a set $P$ is a set of ordered pairs of elements from $P$, $(p, q) \in E$, satisfying the following properties:

1. $(p, p) \in E$,
2. $(p, q) \in E \Rightarrow (q, p) \in E$, and
3. $(p, q) \in E$ and $(q, r) \in E \Rightarrow (p, r) \in E$.

The equivalence relation establishes a very strong constraint for validity; often a less constraining criteria is acceptable. Another useful relation that preserves the notion of closeness is the metric relation. First the term metric is defined and then used to develop the definition of metric relation.

**Metric.** Let $P$ be a set with elements $p, q,$ and $r$. A metric is a function $d:(P \times P) \rightarrow (R)$,

$R = \text{real numbers, satisfying the following properties:}$

1. $d(p, p) = 0$,
2. $d(p, q) = d(q, p)$,
3. $d(p, q) = 0 \Rightarrow p = q$, and
4. $d(p, q) + d(q, r) \geq d(p, r)$.

**Metric Relation.** Let $P$ be a set with metric $d$. A metric relation $R$ is a set of ordered pairs of elements $(p, q)$ from $P$ such that $d(p, q) \leq \delta$ . Points $p$ and $q$ satisfying $(p, q) \in R$ are said to be related under metric $d$ with parameter $\delta$.

It is now possible to present a formal and quantitative definition for validity.

**Validity.** Let $\beta$ be a natural system that is to be investigated using simulation. Let $M^*(S^* \times I) \Rightarrow (S^*)$ be a perfect model for $\beta$ and let $M:(S \times I) \Rightarrow (S)$ be a model for $\beta$.

Then, $1.$ Model $M$ is said to be valid under equivalence if there exists an equivalence relation $E$ such that $L(M^*) \approx_{E} L(M)$, and

$2.$ Model $M$ is said to be valid under metric $d$ with parameter $\delta$ if there exists a metric relation $d$ such that $L(M^*) \approx_{d} L(M)$.

An example is presented to illustrate the formal definition of validity. Let the system shown in Figure 4 be the natural system. In addition, for the application of interest, suppose that the control signal will be set to 1 so the count function of the system is always enabled. Then the perfect model $M^*$ for this system is the function defined in Table 1 when the rows in the table corresponding to Control = 0 are deleted.

Now suppose a new model $M$ for the Mod2 Counter is developed that has no control; the count function is always enabled. This model is shown in block diagram form in Figure 7 and a tabular definition for function $M$ is given in Table 2. The associated labeled transition system $L(M)$ is shown in Figure 8.
Table 2. Definition of M for Mod2 Counter Model

<table>
<thead>
<tr>
<th>State</th>
<th>Signal</th>
<th>N. State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 8. L(M) for Mod2 Counter Model

The labeled transition systems L(M*) and L(M) are shown together in Figure 9. In this simple example, it is easy to observe that the relation E = {(0, 0), (1, 1)} is a bisimulation. Further, since the corresponding state values of related nodes are equal, E is an equivalence relation. Thus, we conclude that M is a valid model for M* under equivalence. This means that if the two models were simulated for any initial state and any sequence of inputs, the model behaviors would be equivalent.

Figure 9. Bisimulation for L(M*) and L(M)

MODEL COMPOSITION

Here we focus on simple function composition of the form G \circ F. This is done for two reasons: first, it leads to a simple explanation of our approach to establishing validity of composed models; and second, it is the basis for developing a general approach to establishing validity of composed models since all function composition can be expressed in terms of simple composition. Suppose F: X \rightarrow Y and G: Y \rightarrow Z are functions. Then the composition of F with G, denoted G \circ F, is the function defined by z = G[F(x)]. It should be noted that the composition exists only for values of x in the domain of F such that F(x) is in the domain of G. Pairs of functions or models for which this occurs are said to be compatible. The composition of two functions is displayed graphically in Figure 10.

Figure 10. Composition of F with G

Because models have been defined as computable functions, composition of models is just composition of functions, which is a well-defined mathematical concept. The informal definition of composability given in the Introduction conveys the intent of composability from a practical point of view, but is insufficiently formal to support a theory of semantic composability. Because models are computable functions and it is known that the set of computable functions is closed under composition (Davis, 1994), any set of compatible models can be composed. However, there is no guarantee that the resulting composite function is a useful model. This is an important point; our formal definition of models as computable functions exhibits the same distinction between syntactic and semantic composability. Interestingly, however, it reduces the matter of syntactic composability to the determination of whether the functions are compatible. If the two models are compatible, the resulting composite function is also a model. The focus of composability in the theory then becomes semantic composability, the question of whether the composite model is valid. The theory will be interested in whether properties of computable functions, such as validity, are preserved in composition. This leads to a new formal definition for composability.

Composability. A set of valid models is composable if and only if the composition is valid.

Intuitively, the validity of a simple composition of models is explained as follows. Suppose F is a valid model for the perfect model F*. Then, there is a non-empty subset of the domain of F, \tilde{X} \subseteq X, where model F is a valid model for F*. Let \tilde{Y} \subseteq Y be the image of \tilde{X} under F. In a similar manner, let G be a valid model for the perfect model G*. Then there is a non-empty subset of the domain of G, \tilde{Y} \subseteq Y, where model G is a valid model for G*. Now, the composition is valid if the valid range of F, \tilde{Y}, and the valid domain of G, \tilde{Y}, have points in
common; that is, there exists a non-empty set \( Y = \bar{Y} \setminus Y \). If this set exists, then the composed function \( G \circ F : \bar{X} \rightarrow \bar{Z} \), where \( \bar{X} \) is the pre-image of \( \bar{Y} \) under \( F \) and \( \bar{Z} \) is the image of \( \bar{Y} \) under \( G \), is a valid model for \( G' \circ F' \). The objective is to develop a formal procedure, using the quantitative characterization of validity developed in the previous section, for testing validity under model composition. Of particular interest is this question: given two valid models, is their composition valid?

We first consider the validity under an equivalence relation of the simple composition of two valid models. A similar approach can be applied to validity under a metric relation and to more complex forms of model composition. Let \( H' = G' \circ F' \) be a perfect model and let \( H = G \circ F \) be a model formed by the simple composition of two valid models. Since \( H' = G' \circ F' \), there exists an equivalence relation \( R_E \) such that \( L(H') \approx_{R_E} L(G' \circ F') \); similarly, since \( H = G \circ F \), it follows that \( L(H) \approx_{R_E} L(G \circ F) \). These two bisimulations are shown in Figure 11. Now, since \( G \) is a valid model under equivalence for \( G' \), it follows that there exists an equivalence relation \( R_G \) such that \( L(G') \approx_{R_G} L(G) \). In addition, since \( F \) is a valid model under equivalence for \( F' \), there exists an equivalence relation \( R_F \) such that \( L(F') \approx_{R_F} L(F) \). These additional bisimulations also are shown in Figure 11. Using Property 3 of the Equivalence Relation Definition, it is easy to show that there exists an equivalence relation \( R \) such that \( L(H) \approx_R L(H') \). It follows that \( H \) is a valid model for \( H' \) under equivalence.

VALIDITY UNDER COMPOSITION

Generalizing the question of validity under composition, an important purpose of semantic composability theory is to establish the validity of compositions of models for different classes of models and validity relations. For some classes of models and relations, it is possible to prove that validity is preserved when valid models are composed (the previous section is a simple example). Clearly, results of this type could be of considerable value in practical applications of composability. Ultimately, it is hoped that the theory leads to the formulation of a software development environment that supports the construction of valid models though composition from a library of valid components.

Classes of models being studied initially include linear functions, affine functions, algebraic functions, elementary functions, and computable functions; classes of relations include equivalence relations and several different metric relations. Current results from this work are tabulated in Table 3. A “Yes” in Table 3 indicates that a composition of valid models from the function class indicated by the column label is provably valid under the relation indicated by the row label, whereas “No” indicates that such preservation of validity cannot be proven for those classes. “Cond.” indicates that there are conditions for which preservation of validity holds and “Conj.” indicates that the result shown is conjectured but not yet proven. We expect that additional function classes and relations will be identified as the theory is applied to the development of component libraries for specific application domains.

<table>
<thead>
<tr>
<th>Function Class</th>
<th>Linear</th>
<th>Affine</th>
<th>Algebraic</th>
<th>Computable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalence</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Metric</td>
<td>Yes Cond.</td>
<td>Yes Cond.</td>
<td>No Conj.</td>
<td>No Conj.</td>
</tr>
</tbody>
</table>

COMPUTATIONAL COMPLEXITY

There are a number of problems associated with composability whose computational complexity is of interest. Four such problems are listed here; they are formulated relative to components, which are models implemented for use in simulation systems.

1. Given a repository of components, how difficult is it (i.e., what is the computational complexity of an
algorithm) to find a subset of the components to compose that meets a given set of requirements?

2. Given a component and a formal meta-model describing its semantics, how difficult is it to determine what specific requirements it meets, or equivalently, how difficult is it to determine if a component meets a specific requirement?

3. Given a set of valid components to compose and formal meta-models describing the semantics of each, how difficult is it to determine if a composition of the components is valid?

4. Given a set of components to compose, each with known execution time complexity, what is the execution time complexity of their composition?

The computational complexity of the first problem, which is called component selection, has been resolved. In earlier work, Page and Opper define four variants of the component selection problem, which differ in terms of whether or not a composition of components meets requirements that none of the individual components does and whether or not requirements that each component meets could be determined efficiently (Page, 1999). (The latter issue is regarded as a separate problem here.) They give a proof that one variant of the component selection problem is NP-complete. More recently, we define two additional variations of the problem which allow for the possibility that a composition might not meet all the requirements met separately by its components, and then define a general form of the problem that subsumes all six variants of the problem. We prove the general problem to be NP-complete, even if it is assumed that the requirements met by each component are known in advance (Petty, 2003c). The computational complexities of the other three problems remain open, though conjectures have been given (Page, 1999), (Petty, 2003c).

CONCLUSIONS AND FUTURE WORK

Composability is an important concept in modeling and simulation. The basic concepts of a formal theory of semantic composability include formal definitions for model, simulation, validity, and composition. The theory is used to investigate the validity of models formed from the composition of valid simulation components. Some results identifying classes of models and validity relations for which validity is preserved and the computational complexity of composability-related problems are presented. A theory of composability can facilitate the convenient reuse of simulation components, which holds the potential to reduce significantly the time and cost of simulation development. More detailed and formal explanations of semantic composability theory are available (Petty, 2003a), (Petty, 2003b), (Petty, 2003c), (Weisel, 2003). Our future work will focus on two areas. First, we hope to investigate validity under composition for additional classes of functions and validity relations. This effort will expand and enhance the results presented in Table 3. Second, we plan to investigate the development of semantic meta-model formalisms. Meta-models are descriptions of models; semantic meta-models are descriptions of model semantics. We hope to demonstrate that semantic meta-models can be used to determine if the semantics of a composition of models are valid.

ACKNOWLEDGEMENT

This research was sponsored by the Office of Naval Research and the Defense Modeling and Simulation Organization under Contract N00014-03-1-0089. That support is gratefully acknowledged.

REFERENCES


