Proceedings of the 1991 Winter Simulation Conference
Barry L. Nelson, W. David Kelton, Gordon M. Clark (eds.)

THEORIES OF DISCRETE EVENT MODEL ABSTRACTION

Suleyman Sevinc

Department of Computer Science
University of Sydney
NSW 2006 AUSTRALIA

ABSTRACT

It is known in systems science that many consistent models at different levels of detail exist for a given modellee. Model abstraction is concerned with identifying relationships between such models and with processes of deriving more abstract ones from more detailed ones. This article critically examines possibilities for a formal basis of model abstraction and possibilities for procedures to perform model abstraction.

1 INTRODUCTION

1.1 What is Model Abstraction?

What can be said about the relationships[1] between the users' and the technical manuals of a computer system? Obviously, they both must represent the same system at different levels of detail. Similarly, discrete event simulation models may be simplified or made more detailed for a less detailed/detailed analysis of the system being studied. Consider a computer network model. For a detailed study such a model may be so constructed that it simulates processing of every **bit**, e.g., transmission of every bit would be a separate event. The same model may be simplified in such a way that the resulting simplified model may only simulate **packet** level events, e.g., instead of simulating transmission of every bit the new model simulates transmission of every packet which is a sequence of bits. When two such models represent the same reality at different levels of detail, the one with less detail is said to be more abstract than the other. In the example above, the model with events at packet level is a more abstract one.

It is important to note that truly abstract models will not, in general, produce the same exact behavior as their more detailed equivalents. This is due to some knowledge loss in the process of abstraction. In the case of the above example, a computer network model operating at packet level would generate a slightly different behavior from the one operating at bit levels. In general, any event that happens which requires time scale smaller than that of a packet, such as collisions during he transmission of a packet, could only be represented with some approximation. If a more abstract model generates a behavior which is close enough to that of its more detailed counterpart, it should be a **valid abstract** model. Unfortunately, however, being close enough in behavior has not been defined for discrete event models and therefore the previous sentence needs to be explained.

Both Foo (1974) and Zeigler (1976) indicate that the model abstraction relationship **preserves some essential features of the original model.** In other words, while certain parts of the model are kept invariant other parts are changed somewhat arbitrarily. Parallel to this, the behavior of the model also shifts around the parts that stay unchanged (or changed slightly). This may be a satisfactory explanation when considering pairs of models but may not explain the validity of more abstract models relative to the original model. In this article, we will not attempt providing an exact definition for model abstraction term. Instead, we will explore issues that we think will be helpful in understanding model abstraction.

So far in our discussion the abstraction relationship is restricted to be between discrete event simulation models. Discrete event simulation models are typically designed to replicate the behavior of their real counterpart and by definition are dynamic and show a time dependent behavior. There are, however, other types of abstractions from discrete event system models to causal models or to qualitative models. Such models still preserve some essential features of the discrete event models that they were derived from, but they are slightly different from discrete event

1. see Wymore, 1986

models in spirit.

1.2 Why Model Abstraction?

Model abstraction, in general, serves two purposes; a)it increases our understanding of models and model behavior b)it may provide us with computationally more efficient models of systems we study. Model abstraction, although done by modellers quite often, is not a very well understood subject. We can identify abstract models in straightforward cases such as in the computer network example used before. By straightforward we mean the abstraction relationship is clear which in that case is **packets** vs. **bits**. The question "are there intermediate abstract models between bit-level and packet-level models?" is a legitimate one and the answer would be of great intellectual value.

1.3 The Previous Work on Model Abstraction

There is not a huge literature on model abstraction, at least not in the way we describe it in this article. Innis and Rexstad (1983) identify a number of simplification techniques and provide some useful insight to simplification and its procedures. They suggest that brevity, transparency and efficiency as three criteria for simplicity. They also provide a guide to be used in simplification. While their research is interesting in that it may help a modeller to go about simplification in a somewhat formulated way, they do not attempt formalizing simplification.

Fishwick (1988) also proposes a number of methods for abstracting processes. These methods combined with definitions of a process at different levels is suggested to form a partially ordered graph called an "abstraction network" which in turn is used to study the model behavior at different levels of detail. A simulation environment HIRES is reported to have been developed to support this process.

Courtois (1985) presents an intuitive argument about reducing complexity of systems in order to be able to study their behavior more efficiently. Several arguments are raised about situations which may be delicate to simplify.

Simplification finds its formal roots, as we understand it, in systems theory. Wymore (1986) concentrates on the question of (notation independent) equivalence between systems. He describes homomorphism as the basis for such a relation and considers homomorphic images of systems as consistent simplifications. Foo (1974) also maintains that homomorphism as being the formal basis for simplification. He raises the question of validity of simplification procedures. He proposes that validity of simplification procedures be determined in terms of properties to be preserved. He also raises several interesting issues such as conversion between stochastic and deterministic systems and approximate homomorphisms

Zeigler (1976) is similar to Wymore's and Foo's approach. In later sections, his approach will be studied more carefully.

Sevinc (1990a) is based on a modified definition of homomorphism and describes a set of programs that generate simplified versions of models from simulation runs.

2 THEORIES OF MODEL ABSTRACTION

2.1 What is a Theory of Model Abstraction?

Model abstraction identifies a relationship between two models. A theory of model abstraction is a well-defined formal expression of what such a relationship is. It does not attempt identifying ways of abstracting models. Given two models, the theory should tell whether they are related via this abstraction relationship or not.

2.2 Possible Theories of Model Abstraction

Wymore (1986) proposes homomorphisms as a formal basis for model abstraction. Zeigler (1976) also suggests homomorphisms as a formal basis. Zeigler's argument is based on his theory of discrete event systems (DEVS). A single component DEVS is described as following;

$$M=<X,Y,S,\delta_{ext},\delta_{\Phi},\lambda,t_a>$$

where, X, Y and S are input, output and state sets, respectively. In DEVS, another component Q, called the total state set is used. The total state set associates the interval $[0, t_a(s)]$ with each s in S, where t_a is the time advance function, which maps S to the non-negative real number set including infinity. λ is the output function which maps Q into Y. The transition function is modularized into external and internal transition functions. The external transition function maps the Cartesian product of Q and X to S. The internal transition function is defined from $S \rightarrow S$.

$$M=<X_M,S_M,Y_M,\delta_M,\lambda_M,t_M>$$

and

$M'=<X_{M'}, S_{M'}, Y_{M'}, \delta_{M'}, \lambda_{M'}, t_{M'}>$

both legitimate DEVS models, where

$X_M=X_{M'}$,

$Y_M=Y_{M'}$.

A DEVS homomorphism from M to M' is a map h such that

1) $h:S_M \rightarrow S_{M'}(onto)$

2) $h(\delta_M(s,e,x))=\delta_{M'}(h(s),e,x)$

3) $\lambda_{M'}(h(s),e)=\lambda_M(s,e)$

4) $h(\delta_{M,\Phi}(s))=\delta_{M,\Phi}(h(s))$

5) $t_{aM'}(h(s))=t_{aM}(s)$

where $x \in X$ and $(s,e) \in Q_M$

Zeigler in fact proves that this definition of homomorphism preserves behavior, however, it can not fully account for abstraction in the way we understand it. Consider the computer network example from the previous sections. Typically a sequence of states lumped together to extract packet-level behavior which would require adding of time advance values for microstates to extract the time advance value for more abstract packet-oriented states. The definition of homomorphism completely rules this out. Many times mapping of time advance values involves an intermediate function which takes time advance values of microstates in and generates a new time advance value for the abstract state.

Sevinc (1990a) proposes a weakened definition of homomorphism to be used as a formal basis for model abstraction as briefly explained in the following:

1) $h:S_M \rightarrow S_{M'}$(onto)

2) $\delta'(h(s),e,x)=h(\delta(s,e,x))$
with $p_{x,e}(h(s),h(\delta(s,e,x)))>0$

3) $\lambda_{M'}(h(s),e)=\lambda_M(s,e)$ *with* $p_s>0$

4) $\delta'(h(s))=\{h(\delta(s))$ *with* $p(h(s),h(\delta(s)))>0\}$

5) $t_a(h(s))=U\{t_a(s') \mid h(s')=h(s)$ and
for *every* s, $t_a(s) \geq t_a(s')$,
$t_a(s'') \mid h(s'')=h(s)$ and for *every* s,
$t_a(s) \leq t_a(s'')\}$

Where $x \in X$ and $(s,e) \in Q_M$. $p_{x,e}(s_1,s_2)$ and p_s are probabilities computed from the observations of the behavior of the original model. 'h' lumps a number of states together, therefore in order for the simplified model to make a transition from one state to another the original model must have done such a transition during the observation period. This fact is stated in item 4 above by requiring a non-zero probability for such a transition. Since a number of states are lumped together in the simplified model, a number of outputs are possible at a particular simplified state. This conflict is resolved using p_s in item 3 above. For external transitions of the simplified models similar probability values are used from the observations except that they are conditioned on the type of input and elapsed time values which are also simplified. Time advance values for simplified models are approximated using random values distributed uniformly over the values between the minimum and maximum time advance values which are mapped to a particular simplified state.

Basically, the existence of a detailed model is assumed and observed in an experimental frame on the basis of which the simplified version is constructed. Typically, a modeller starts with step 1 by defining an 'h'. What needs to be achieved is $P\{|\Xi(q, \omega) - \Xi'(h(q), \omega)| \leq \varepsilon\} \geq \delta$, for every $q \in Q$ and $\omega \in \Omega$; Ξ and Ξ' are either state or output trajectories of the original and the lumped models, respectively. The modeller may have to try several h's before he can ensure the above inequality for δ and ε values he chooses.

This weakened definition of homomorphism still suffers from the same symptoms as strict homomorphism does. Even if we allow the use of an intermediate function for time advance values such as f(s1,..,sn) where $h(s_i)=S$, the definition still is not able to account for a wide variety of cases. Consider the computer network example used before. The abstraction is not done in a way that a set of micro states are mapped to an abstract state but a sequence of states is mapped to an abstract state, i.e., a packet transmission corresponds to transmission of a sequence of bits. Therefore, abstraction relationship should include this possibility in its definition. In fact many different sequences would qualify to be a packet, thus a packet referring to a set of microstate sequences each one with possibly distinct features. For example the total

time required to transmit a packet may vary from packet to packet although this time may be fixed for individual bits. Furthermore a packet transmission may have to be incomplete due to some factors such as congestion. Therefore, the definition of abstraction relationship should allow inferring time related properties of packets from what has been observed from simulation runs at bit level. An observation about time dependent behavior of more abstract models shows that more detailed models have a finer time scale, that is they have to process more events to cover the same period of simulated time. In fact this may be part of the definition of an abstraction relationship.

Another observation of abstract model behavior reveals that their inputs and outputs also have to be lumped together towards more abstract input and output events. This might eventually affect the input/output compatibility assumption made in Sevinc (1990a) such that abstraction of a part of a system may necessitate abstraction of the rest of the system for compatibility.

The conclusion of this section is that a globally valid definition of abstraction is lacking. We are not able to define abstraction in its most general sense at the moment. In the limited context of the computer network example used above; if a procedure can be defined to derive the protocols of a computer network from its models operating at bit levels then it should be considered a true abstraction. At least, this is the sense in which we are trying to study the concept of model abstraction.

3 PROCEDURES FOR MODEL ABSTRACTION

3.1 What is a Procedure of Model Abstraction?

A procedure for model abstraction is basically an algorithm describing how to derive more abstract models from more detailed simulation runs. This is a bit different from constructing state machines from their input/output behavior in that an abstraction procedure typically has access to states and state changes of the detailed models it operates upon. You may be pondering, "How it is possible to construct a procedure for something we do not have a precise definition for?". Consider the field of induction or belief revision where the subject is only partially understood but many procedures have been developed to operate under various restriction. Abstraction procedures are much like that; they make use of the partial knowledge we have about abstraction and therefore they are partial solutions that are valid for subclasses of abstraction under specific assumptions. In fact, I would go a step further and suggest that the procedures themselves sometime may not be well-understood and it should not prevent us from experimenting with them. As in the fields chemistry and physics, sometimes theories follow the laboratory experiments. Therefore, it makes sense to talk about procedures for model abstraction at this stage.

3.2 Procedures for Model Abstraction

The most straightforward theory-based partial abstraction procedure would be generating homomorphic images from model specifications. Take Zeigler's definition of homomorphisms as an example. If we can identify group of micro states with equal time advance values, transition and output characteristics we could construct a notationally independent machine which would be a more abstract model. Surprisingly enough, although the procedure is simple enough, no one, to my knowledge, has experimented with such a procedure.

Another procedure described in Sevinc (1990a) which requires an initial knowledge of mapping characteristic sets of a model to more abstract ones. A set of agents then observe the model behavior to determine the transition, output and time characteristics of these abstract states. A code generator generates a model expressed in a canonical form. Experimental results show that the simplified models reduce the simulation run time considerably while preserving the behavior within certain limits. The limitation of the procedure is that it does not have the ability to identify more abstract models without an initial knowledge which must be provided by the modeller. The procedure is the first attempt we know of to automate the model abstraction process.

A second abstraction procedure is reported in Sevinc (1990b). The procedure extracts logical statements from discrete event model states. The logical statements derived for adjacent states, i.e., related via transition, are used to identify events by measuring their differences in terms of statements turned on and turned off. A planning algorithm is then used to answer queries about simulation model behavior. The procedure abstracts simulation models to a logical model where the power of deduction and planning is used. This is not an abstraction process whose outcome is an executable discrete event simulation model. Logical equivalents of discrete event models then can be used to learn and reason about the simulation model behavior.

Perhaps the most interesting procedure described in the literature is the one described in Sevinc and Foo (1990). The procedure assumes the existence of a database which stores distances between states of a model. The assumption is that the closer the states to each other the more likely it is that they are semantically similar. In reality, similar states are likely to follow each other when the change is not a dramatic one. The method uses a classification technique to find clusters of microstates and each cluster is represented by an abstract state whose properties are derived from those of the micro states in the cluster. A second part of the algorithm is aimed at error correction. It is recognized that clustering algorithms may occasionally misplace the micro states. The second part of the algorithm, using the transition characteristics of the micro states and clusters, locates such misplacements and attempts placing these micro states in clusters which more closely match their transition characteristics. The success of this procedure heavily depends on the distance metric used to measure the distance between micro states. The strength of the method is that it is fully automated and that it is capable of generating abstract models at any desired level. Furthermore, it can also address the case where a sequence of micro events being mapped to an abstract event in a limited way. It is also capable of generating abstractions that may not have been foreseen by the modellers themselves.

4 CONCLUSIONS

Model abstraction is frequently done by modellers when the complexity of their models exceed the power of their resources, i.e., too much space or time would be taken by simulations. It is the responsibility of simulation methodologists to propose sound ways of generating more abstract models from more detailed ones. Model abstraction also serves the purpose of learning more about model behavior and inferring properties that otherwise may remain uncovered. No complete theories of model abstraction exist, nor does any sufficiently general procedure. The field, with only less than half a dozen published articles, is wide open inviting the attention of simulation methodologists.

5 REFERENCES

P.J. Courtois. 1985. On Time and Space Decomposition of Complex Structures. *Communications of the ACM*, Vol. 28, No. 6, pp. 590-603.

P.A. Fishwick. 1988. The Role of Process Abstraction in Simulation. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 18, No.1, pp. 18-39.

N.Y. Foo. 1974. Homomorphic Simplification of Systems. *Doctoral Dissertation, University of Michigan.*

G. Innis, E. Rexstad. 1983. Simulation Model Simplification Techniques. *Simulation*, July, pp. 7-15.

S. Sevinc. 1990a. Automation of Simplification in Discrete Event Modelling and Simulation. *International Journal of General Systems*, **18**, No. 2.

S. Sevinc. 1990b. Extracting Logical Theories From DEVS Models, Tech. Rept. 377, Basser Dept. of Computer Science, University of Sydney.

S. Sevinc and N.Y. Foo. 1990. Discrete Event Model Simplification via State Classification. In: *AI and Simulation Theory and Application*, edited by W. Webster and R. Uttamsingh, SCS, pp.211-216.

A.W. Wymore. 1986. *A Mathematical Theory of System Design.* SANDS, Tucson, Arizona.

B.P. Zeigler. 1976. *Theory of Modelling and Simulation.* John Wiley, New York.

6 AUTHOR BIOGRAPHY

SULEYMAN SEVINC is a lecturer in the Department of Computer Science at the University of Sydney, Australia. His research interests are model abstraction and theory-based modelling environments.