Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

CONTROL FLOW GRAPHS AS A REPRESENTATION LANGUAGE

Bruce A. Cota
Douglas G. Fritz
Robert G. Sargent

Simulation Research Group
449 Link Hall
Syracuse University
Syracuse, New York 13244

ABSTRACT

A model representation language called Control Flow Graphs whose purpose is to explicitly provide the information needed for simulation execution algorithms for different types of computer architectures is briefly described. An overview of a hierarchical modeling language containing two types of hierarchical structures for use with the Control Flow Graph representation language is given. Also, the properties of a model representation language are presented and how Control Flow Graphs are related to these properties is discussed.

1. INTRODUCTION

Control Flow Graphs were developed by Cota and Sargent (1989, 1990a, 1990b, 1990c, 1990d) as a model representation language for discrete event simulation. The purpose of this representation language is to make information explicit to enable the development of simulation execution algorithms for parallel and distributed computers. This avoids requiring the modeler to add additional information as is usually required, e.g., lookahead information. Simulation execution algorithms for Control Flow Graphs have been developed for sequential and parallel/distributed computers.

A hierarchical model language that can automatically generate a Control Flow Graph representation has been developed by Fritz and Sargent (1993). While a representation language can be used for modeling, it is not designed for that purpose and thus may not be "user friendly" for modeling. The Control Flow Graph representation is straight forward to use in the modeling of simple systems but can become very complex when modeling complex systems. The hierarchical modeling language addresses this complexity. Figure 1 shows the relationships between the hierarchical modeling language, the Control Flow Graphs representation language, and the simulation execution algorithms.

The remainder of this paper is organized as follows: Section 2 describes Control Flow Graphs, Section 3 presents the hierarchical modeling language, Section 4 gives the properties of model representation and discusses how Control Flow Graphs are related to them, and Section 5 summarizes the paper.

2. CONTROL FLOW GRAPHS

Control Flow Graphs are based upon a new theoretical foundation for discrete event simulation called the "Modified Process-Interaction World View" (Cota and Sargent 1992). In this new foundation the process-interaction world view (Zeigler 1976) was modified so that each model component (or logical process) has the modularity property (encapsulation and locality) and interacts with other components only through a strictly defined interface. One such interface is message passing which is the method used by Control Flow Graphs. In addition, the modified process-interaction world view uses the active receiver model (Cota and Sargent 1992) which means that the component (or process) determines when to react to information (e.g., messages) received from other components (as contrasted to the passive receiver model - used, e.g., in object-oriented programming and simulation - where components react when information is received). Furthermore, the modified process-interaction world view favors the active resource world view over the more widely known active transaction world view (Henrikson 1981) used in many simulation languages such as GPSS.

In the Control Flow Graphs representation, the behavior of each system component (or process) is specified by a Control Flow Graph and the interactions between components are specified by message passing over directed channels interconnecting model components. Each channel carries only one type of message which implies that there may be multiple channels between two components. Messages queue on

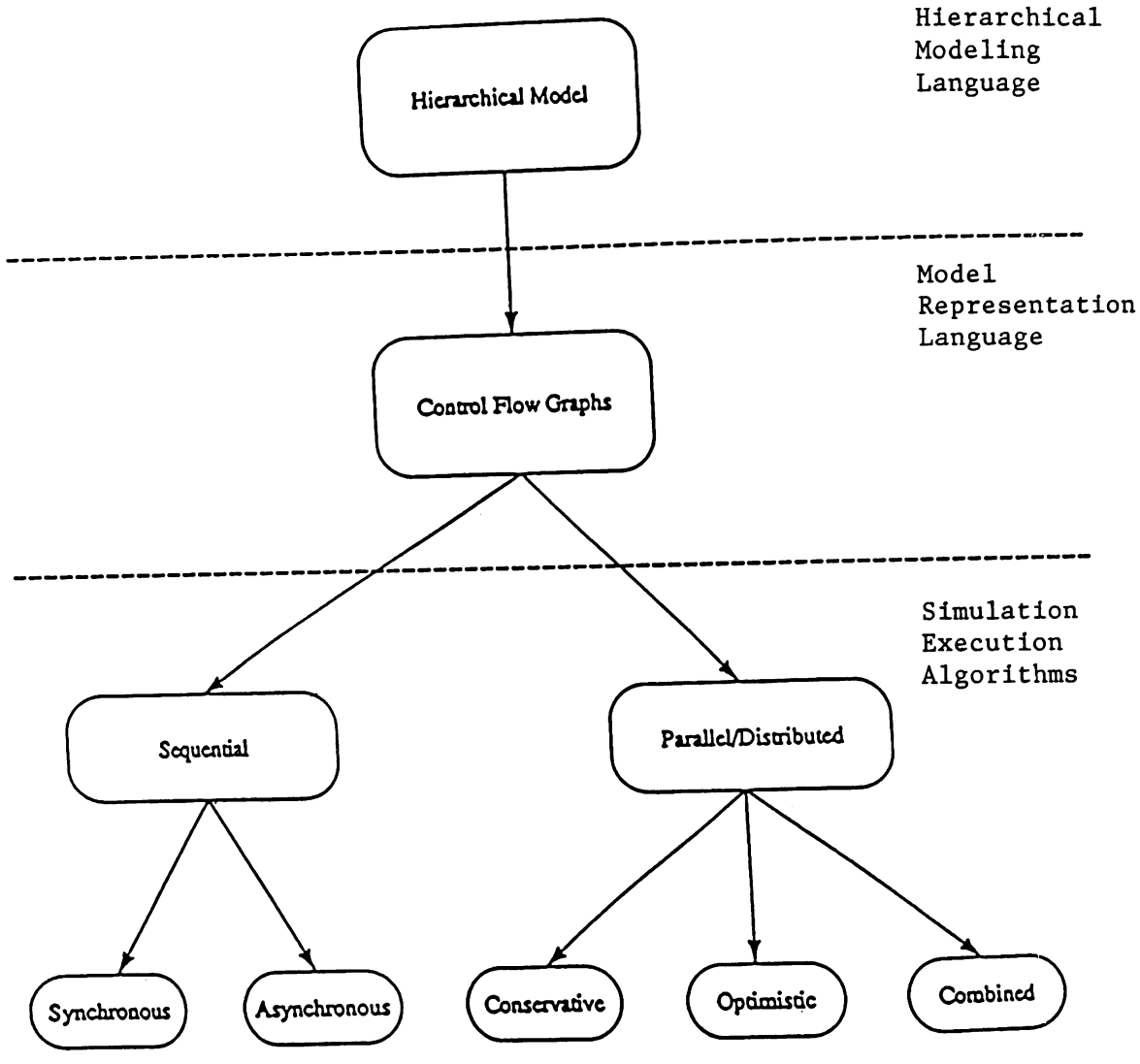

Figure 1. Hierarchial Modeling and Simulation System

each channel until the Control Flow Graph describing the component decides to receive them, i.e., Control Flow Graphs are active receivers. The time-stamps on the messages are their sending times. (This is in contrast to the method generally used in parallel and distributed simulation in which the timestamps are to the times the messages are to be received.) The specification of channels is accomplished via an Interconnection Graph. An Interconnection Graph is a directed graph where the nodes represent components (Control Flow Graphs) and the directed edges represent the channels. Thus, one Interconnection Graph and a set of Control Flow Graphs (one for each system component (or process)) are required to specify a simulation model using the Control Flow Graphs representation.

A Control Flow Graph is an augmented directed graph where nodes represent the control states of a component and edges specify possible component control state transitions. A control state is a formalization of the "process reactivation point" (Cota and Sargent 1992 and Zeigler 1976). Each edge has three attributes: an event, a condition, and a priority. The event specifies the actions to be taken if that edge is transversed which includes receiving a message waiting at a channel, changing values of local (model component) variables, and sending messages over channels. The condition specifies under what conditions an edge can be considered for traversal. There are three types of conditions: (i) a channel has messages waiting to be received (i.e., is nonempty), (ii) a Boolean expression on the values of

the local variables, and (iii) a time delay condition which becomes true after the passage of some specified local (component) simulation time. Each Control Flow Graph has its own (local) simulation clock. The edge selected for traversal from a node (control state) is the edge that is true with the lowest local simulation time. In case of time ties, the tied edge that has the highest priority is selected.

Each Control Flow Graph has its own "thread of control" - i.e., it operates "independently" of other Control Flow Graphs (except for message passing interaction). A "Point of Control" in each Control Flow Graph moves from control state to control state. At each control state, the point of control may wait for a period of (local) simulation time, or move immediately if a Boolean condition on local variables is true or if it is to receive a waiting message. The decision as to which edge to traverse may have to be delayed in some cases to see if messages arrive from other Control Flow Graphs (components).

Cota and Sargent (1989, 1990a, 1990b, 1990c, 1990d) have developed different simulation execution algorithms for the Control Flow Graphs representation. Two simulation execution algorithms are available for sequential computers (Cota and Sargent 1990b and Fritz and Sargent 1993): synchronous and asynchronous. The synchronous algorithm uses the standard approach of computing the events across all control flow graphs in their time order sequence. The asynchronous algorithm allows events to be executed out of time sequence when this does not effect the simulation results in order to reduce the simulation execution time by eliminating some event list operations. For parallel and distributed computers there are two conservative algorithms (one using null message passing and the other using deadlock detection and resolution), an optimistic algorithm which is more efficient than the usual optimistic algorithm because rollback is based upon the control states of a Control Flow Graph instead of rolling back if a "late message" is received to a Control Flow Graph, and an optimistic Only When Necessary (OWN) algorithm which is a combined optimistic and conservative algorithm. All of these algorithms obtain the information they need directly from Control Flow Graphs and thus a modeler need not add any additional information such as the "lookahead" information required for most parallel and distributed simulation execution algorithms. This is because lookahead is "automatically" obtained by the algorithms from the Control Flow Graphs representation.

3. HIERARCHICAL MODELING LANGUAGE

Fritz and Sargent (1993) have developed a hierarchical modeling language for Control Flow Graphs that uses the same world view as the Control Flow Graphs representation. This modeling language provides for two independent types of hierarchical structures that can be used concurrently. One type of hierarchical structure is the "coupling of components" (Ziegler 1984). The model components are either atomic or coupled components. Coupled components consist of other coupled components and/or atomic components. Atomic components are components whose behavior are specified by Control Flow Graphs (or Hierarchical Control Graphs). In the use of this type of hierarchical structure, one specifies the interconnection (channels) of Control Flow Graphs by a Hierarchical Interconnection Graph instead of an Interconnection Graph. If only atomic components are used, then the Hierarchical Interconnection Graph is an Interconnection Graph.

The other type of hierarchical structure is the use of "Macro Control States" in addition to control states. Macro control states are modular structures that consist of other macro control states and/or control states. This leads to using Hierarchical Control Flow Graphs instead of Control Flow Graphs to specify the behaviors of model components (or processes). (If only control states are used in a Hierarchical Control Flow Graphs, then one has a Control Flow Graph.) The concept of macro control states is very powerful because it provides for partial behavior specifications, recursive decomposition of behavior specifications, higher level model specification constructs to be used, and reuse.

Thus, to specify a simulation model using this hierarchical modeling language, one specifies a Hierarchical Interconnection Graph and a set of Hierarchical Control Flow Graphs (one for each model component). Flattening algorithms have been developed (Fritz and Sargent 1993) to automatically convert a Hierarchical Interconnection Graph into an Interconnection Graph and Hierarchical Control Flow Graphs into Control Flow Graphs. There are several advantages to using these two types of hierarchical structures including reuse (model and software) of components and macro control states, aid to modeling, and aid in model validation. (See Fritz and Sargent (1993) for further details.)

4. PROPERTIES OF A MODEL REPRESENTATION LANGUAGE

In developing a model representation language for discrete event simulation their properties need to be considered. These properties include the purpose of the representation, the requirements that are necessary and desirable, and the level of the representation. The purpose

of our Control Flow Graph representation is to make information explicit in the representation in order to "automatically" obtain the "lookahead" information needed for simulation execution algorithms for parallel and distributed computers. Our requirements were modularity (which aids in modeling, model and software reuse, hierarchical modeling, and parallel and distributed simulation), to be able to have efficient simulation execution algorithms, and to be able to either develop a modeling language with ease or use the representation language as a modeling language.

The level of representation (abstraction) must be such that at least the minimal information required is available. One can use a lower level of representation but this requires more information in the model specification of the representation then is required. For Control Flow Graphs, we specify behavior using control states. As discussed in Fritz and Sargent (1993) higher levels of specification remove capabilities of the Control Flow Graph representation. For example, specifying the state space by "internal" and "external" state sets (a higher level of abstraction) instead of control states removes the ability of simulation execution algorithms to "automatically" obtain "lookahead" but continues to allow the components to be active receivers (see Fritz and Sargent (1993) for definitions and discussion).

Using a model representation requires a modeler to specify the necessary information about a model in some way. This can be directly in the representation or in some modeling language that can be converted into the representation. In our case a modeler may specify a model directly in the Control Flow Graph representation or by using the hierarchical modeling language. The hierarchical modeling language provides the information needed through the Hierarchical Interconnection Graph and the Hierarchical Control Flow Graphs. One interesting aspect about Macro Control States in Hierarchical Control Flow Graphs is that higher level modeling constructs can be developed for Macro Control States that contain the information needed for the representation. Thus if modelers only uses these constructs, they do not need to specify at the level of the representation. This can provide a powerful way of modeling; especially if the modeling language will allow both the use of the higher level modeling constructs (e.g., macro control states) and specification at the level of representation (e.g., control states).

5. SUMMARY

We have briefly described the Control Flow Graphs representation language and a hierarchical modeling language that converts simulation models into this representation. A Hierarchical Modeling and Simulation System (HI-MASS) is under development that uses this representation language, modeling language, and simulation execution algorithms based on the representation. This system eliminates the requirement that a modeler add additional information or model in a specific way depending on the computer architecture on which the simulation is to be executed. This system will allow the same simulation model to run on different computer architectures. Furthermore, two types of hierarchical modeling is allowed by the hierarchical modeling language. This system (HI-MASS) satisfies the requirements of a new model paradigm specified in Sargent (1992).

ACKNOWLEDGMENT

This work was supported in part by the CASE Center of Syracuse University, by the Air Force Office of Scientific Research, Bolling AFB, Washington, D.C., and by Rome Laboratory, Griffiss AFB, N.Y.

REFERENCES

Cota, B.A. and R.G. Sargent. 1989. Automatic lookahead computation for conservative distributed simulation. CASE Center Technical Report 8916, CASE Center, Syracuse University, December 1989.

Cota, B.A. and R.G. Sargent. 1990a. A framework for automatic lookahead computation in conservative *Distributed Simulation*, in Distributed Simulation, D. Nicol, editor, The Society for Computer Simulation, 1990, pp. 56-59.

Cota, B.A. and R.G. Sargent. 1990b. Simulation algorithms for control flow graphs. CASE Center Technical Report 9023, CASE Center, Syracuse University, November 1990.

Cota, B.A. and R.G. Sargent. 1990c. Control Flow Graphs: A method of model representation for parallel discrete event simulation. CASE Center Technical Report 9026, CASE Center, Syracuse University, December 1990.

Cota, B.A. and R.G. Sargent. 1990d. Simultaneous Events and Distributed Simulation, in *Proceedings of 1990 Winter Simulation Conference*, New Orleans, LA, December 1990, pp. 436-440.

Cota, B.A. and R.G. Sargent. 1992. A modification of the process interaction world view, *ACM Transactions on Modeling and Computer Simulation*, Volume 2, Number 2, April 1992, pp. 109-129.

Fritz, D.G. and R.G. Sargent. 1993. Hierarchical Control Flow Graph Models. CASE Center Technical Report 9323, CASE Center, Syracuse University, December 1993.

Sargent, R.G. 1992. "Requirements of a Modeling

Paradigm" (part of panel session on "Discrete Event Simulation Modelling: Directions for the '90's"), in *Proceedings of the 1992 Winter Simulation Conference*, Arlington, VA, December 1992, pp. 780-782.

Zeigler, B.P. 1976. *Theory of Modelling and Simulation*, John Wiley & Sons.

Zeigler, B.P. 1984. *Multifaceted Modelling and Discrete Event Simulation*, Academic Press.

AUTHOR BIOGRAPHIES

BRUCE A. COTA has a B.S. in Mathematics from Buffalo State College and a M.S. in Mathematics from Syracuse University, and is working towards a Ph.D. in Computer and Information Science at Syracuse University. He has been an instructor of Mathematics and Computer Science at Centre College in Danville, Kentucky.

DOUGLAS G. FRITZ is a graduate student at Syracuse University working towards a Ph.D. in computer engineering. His research area is hierarchical modeling for discrete event simulation. He has received M.S. degrees in Electrical Engineering and Computer Science from Syracuse University. He was formerly a development engineer at IBM.

ROBERT G. SARGENT is a Professor at Syracuse University. He received his education at the University of Michigan and has published widely. Dr. Sargent has served his profession in numerous ways and has been awarded the TIMS College on Simulation Distinguished Service Award for long-standing exceptional service to the Simulation Community. His research interests include the methodology areas of modeling and discrete event simulation, model validation, and performance evaluation. Professor Sargent is listed in *Who's Who in America.*