Model update: A model specification formalism with a generalized view of discontinuity

Tuncer I. Ören
Computer Science Dept.
University of Ottawa
Ottawa, Ontario, Canada K1N 9B4

ABSTRACT

A generalized view of discontinuity allows to perceive it as an update of a state variable or the derivative of a state variable. The first update corresponds to a re-initialization of a state variable. The second update corresponds to a modification of a model. Both of the updates are applicable not only to continuous-change models but also to discrete-change models. Memoryless models can have model update via modification of parameter(s). A second level of generalization allows switching among submodels of multi-models.

The formalisms are given for the Gest language with reference to its knowledge based advisor Magest. However, the formalisms are implementable in any advanced model-based simulation environment.

INTRODUCTION

Systems modelled in terms of ordinary differential equations may have two types of discontinuities: discontinuity in a state variable and discontinuity in the derivative of a state variable. Either one or both types might occur at a time instant (Birta et al. 1985; Ören and Ma 1986).

Discontinuity in a state variable corresponds to a re-initialization of a state variable and can also be called re-initialization discontinuity. Since it corresponds to a sudden change of a state variable, it can also be called jump discontinuity.

Discontinuity in the derivative of a state variable, or derivative discontinuity, in short, corresponds to an update of the model. Model update can be done directly or indirecly. Direct update of a model is done by replacing the function which specifies the derivative of a state variable by another function. Indirect update of a model can be achieved by just updating the value(s) of one or several parameter(s) which appear in the derivative function.

When a continuous-change model is simulated during a time interval during which at least one discontinuity of either one or both types occur, then the model is called discontinuous or piece-wise continuous model.

A GENERALIZED VIEW OF DISCONTINUITY

Discontinuity can be conceived from different perspectives. From the implementation point of view, one has two possibilities: In the first case the time of discontinuity is known a priori. In this case, a time event can be scheduled beforehand and at its occurrence, integration is interrupted to perform the required updates after which integration resumes. In the second case, the time of discontinuity is not known a priori. In this case, the detection of a discontinuity triggers a state event and interrupts the integration. The time of discontinuity is refined, the necessary updates are done, and then the integration resumes. From the implementation point of view, piece-wise continuous-change models were considered combined continuous/discrete models (Cellier 1979).

The importance of the implementation of a simulation language should not be denied. However, putting the emphasis on the implementation may distort and limit our perception. For example, designers of early simulation languages for continuous-change models were obliged - due to hardware limitations - to put the emphasis on implementation. Therefore one had to have initial, dynamic, and terminal regions. In this paradigm one has to specify initial values of the state variables before knowing what the state variables are. It has been known since 1971 (Ören 1971) that models should be conceived separately from the experimental conditions and the other codes necessary to drive them under the experimental conditions to generate model behavior.

Contemporary simulation is model-based. Perception of discontinuity as a characteristic of models instead of putting the emphasis on the implementation issues may open new vistas for both traditional and cognizant simulation (Ören 1987). In this article, a generalization of the concept of discontinuity is proposed in terms of two other concepts: updateable models and multimodels.

POSSIBILITIES OFFERED

The possibilities offered are given within the modelling paradigm of Gest87 (Ören and Özmızrak 1987a, b) which is an update of Gest81 (Ören 1984). It is assumed that a knowledge-based modelling and simulation environment such as Magest (Aytaç and Ören 1986) will use these metamodels as meta-knowledge to provide dynamic templates for guidance, advice, and explanation in specifying component as well as coupled models, parameter sets, and experimental conditions.

To represent several dynamic templates, a variant of the unifying graphic representation scheme of Ören for structural algorithms and programs is used (Ören 1984). Since the templates are dynamic, they can automatically be tailored to user requirements by a knowledge-based advisor system.

Updateable Models

An updateable model is a model with an update segment where one can declare update of state variables or parameters and the conditions which should trigger them.

Three possibilities are provided in Gest87: updateable continuous-change model, updateable discrete-change model, and updateable memoryless model. If the optional update section does not exit, one has continuous-change model, discrete-change model, and memoryless model, respectively.

The term discontinuity which is appropriate for piece-wise continuous-change models is not appropriate neither in the case of updateable discrete-change models nor in the case of updateable memoryless models. However, the concept of "updateable model" or "model update" accepts discontinuity as a special case and extends its scope to other types of models.

Updateable Continuous-Change Models. Figure 1 represents the major elements of the dynamic template for updateable continuous-change models. A continuous change model is described by ordinary differential equations. The basic features of the elements are similar to those defined in Gest81 (Ören 1984) and in Gest87 (Ören and Özmızrak 1987a,b) and are not repeated here. The dynamic structure of an updateable continuous-change model consists of three sections:

- derivative section,
- output function section, and
- update section.

In the update section, the update conditions, as well as specifications to update state variables and parameters are declared. For a piece-wise continuous-change model, the update section provides a model-oriented specification of the discontinuities. Figures 2 and 3 are examples to updateable continuous-change models.

The template being dynamic, can be tailored to user needs, by a modelling advisor such as Magest, based on the incremental semantic knowledge provided by the user.

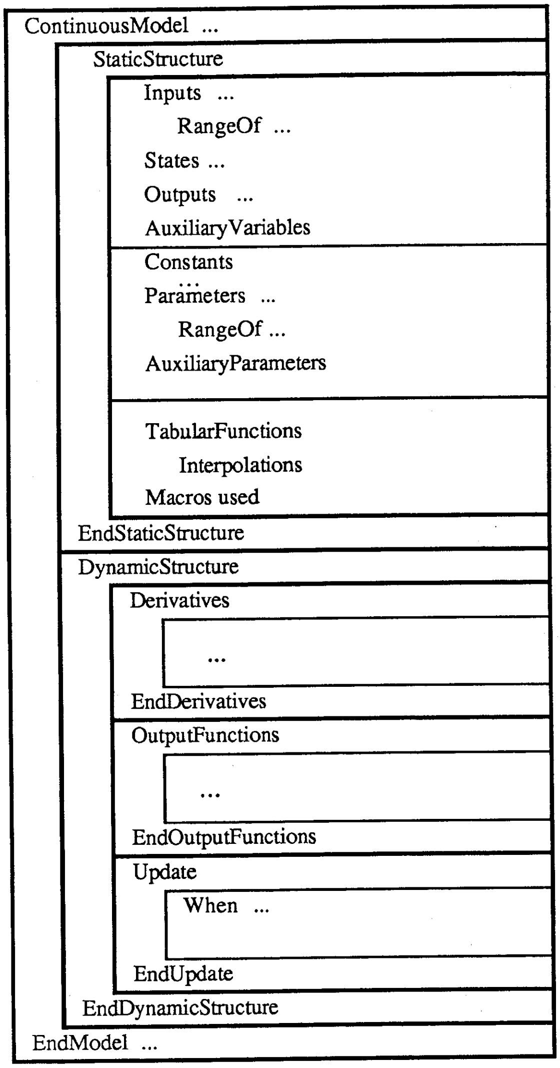

Figure 1. Updateable Continuous Model

Updateable Discrete-Change Models. Figure 4 shows the dynamic template for updateable discrete-change models. A discrete-change model is described in terms of difference equations. The dynamic structure of an updateable discrete-change model consists of three sections:

- state transitions section,
- output function section, and
- update section.

The update section contains specifications of the conditions and the corresponding updates of state variables to re-initialize them and/or update of parameters to update mathematical model representing the system.

```
ContinuousModel Parachute
    StaticStructure
        States      VX, VY
        Outputs     VX, VY.
        AuxiliaryVariable V

        Constant   G
            G = 9.81                          { m/s }
        Parameters  P, M, D
          { P              s  - parachute opening time }
            M  = 70                     { kg  - mass }
            D  = 0.7              { kg/m - drag coef.}
    EndStaticStructure
    DynamicStructure
        Derivatives
            VX' = -(D*V*VX)/M
            VY' = -G-(D*V*VY)/M
            V   =  Sqrt(VX**2 + VY**2)
        EndDerivatives
        Update
            When   Time = P,
                D  =  7.0      { update of parameter }
        EndUpdate
    EndDynamicStructure
EndModel  Parachute
```

Figure 2. A Continuous Model with Update of Parameter

```
ContinuousModel  Bouncing Ball
    StaticStructure
        States      X,  V    { V - Vertical velocity }
        Output      X        { X - Height            }

        Constant   G
            G = 9.81                         { m/s }
        Parameter  K       { Bouncing constant   }
    EndStaticStructure
    DynamicStructure
        Derivatives
            V'  =  - G
            X'  =    V
        EndDerivatives
        Update
            When X Crosses 0.0 (Tolerance = 0.01)
                V = -K*V
        EndUpdate
    EndDynamicStructure
EndModel  BouncingBall
```

Figure 3. A Continuous Model with Re-initialization of a State Variable

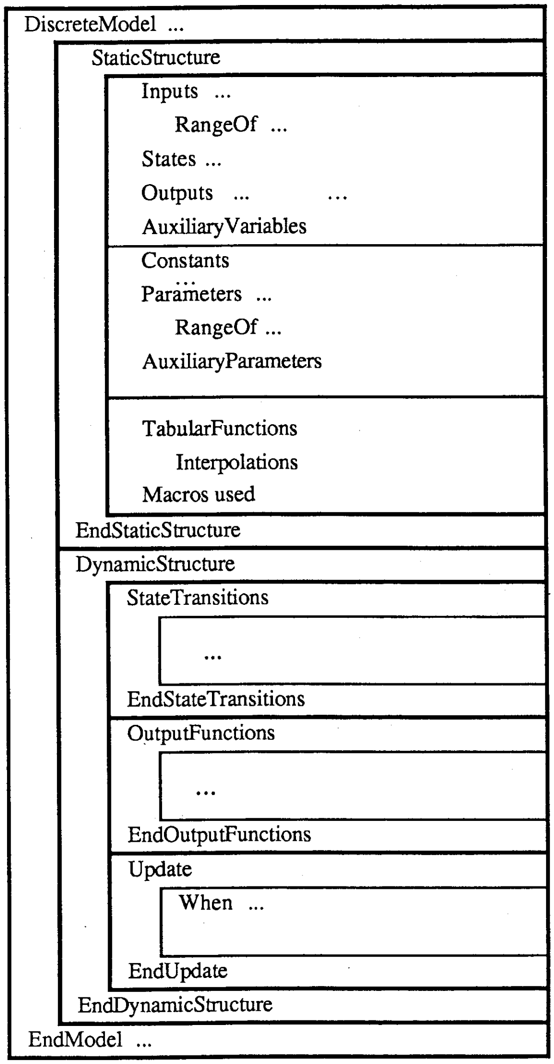

Figure 4. Updateable Discrete Model

Updateable Memoryless Models. Figure 5 is a dynamic template for memoryless models. A memoryless model does not have state variables and transforms instantaneously its inputs into outputs. The dynamic structure of an updateable memoryless model consists of two sections:

- output function section and
- update section.

Since there is no state variable, the update section contains the specifications to update parameters only.

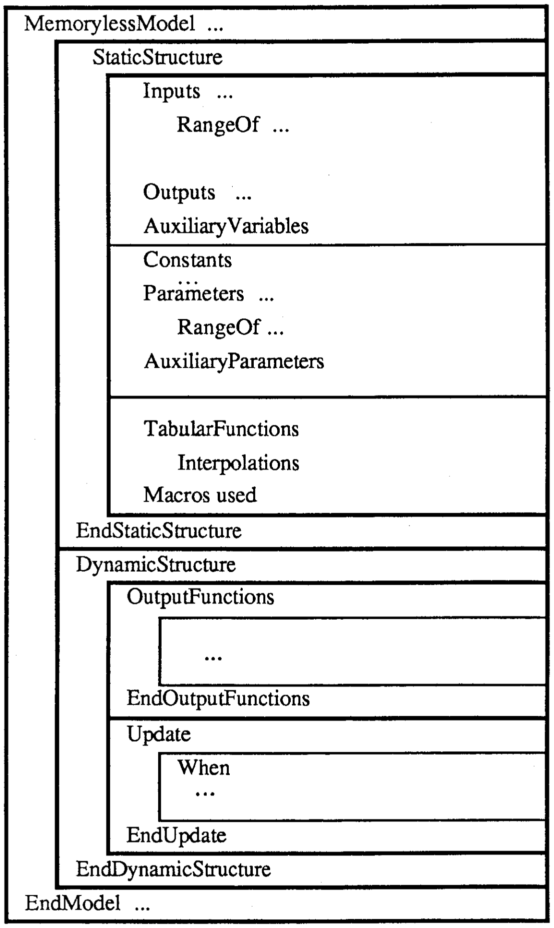

Figure 5. Updateable Memoryless Model

Multimodels

A multimodel is a model which consists of two or more submodels. Only one submodel is active at any time. While a submodel is active, another one can be activated, under pre-defined conditions, to cause a transition (or switching) from a submodel to another one. With the activation of a submodel, one can also optionally re-initialize its state variables and/or parameters.

Three types of multimodels are presented in this article. They are: continuous multimodel, discrete multimodel, and memoryless multimodel.

Figures 6, 7, and 8 represent dynamic templates for continuous multimodel, discrete multimodel, and memoryless multimodel, respectively. One of the dynamic aspects of the template is that it can automatically be tailored for the declared number of submodels.

In a continuous multimodel, for example, the model consists of several continuous-change submodels. While a submodel is active, update conditions are monitored to detect and locate the update time. Then the necessary transition from the current submodel to another one is realized. Optionally, re-initialization of state variables and/or parameters can also be done.

CONCLUSION

The generalization of the concept of discontinuity to updateable models and to multimodels offers useful concepts for both traditional and advanced aspects of simulation.

For the traditional simulation, it provides a clear way of representing problems requiring model update and model switching without confusing model specification issues with model processing issues such as time- or state-events.

For advanced simulation, such as cognizant simulation (Ören 1987), the formalism provides a basis to be incorporated in a knowledge-based modelling advisor to provide guidance and advice to the user and to realize semantic checks of completeness and consistency.

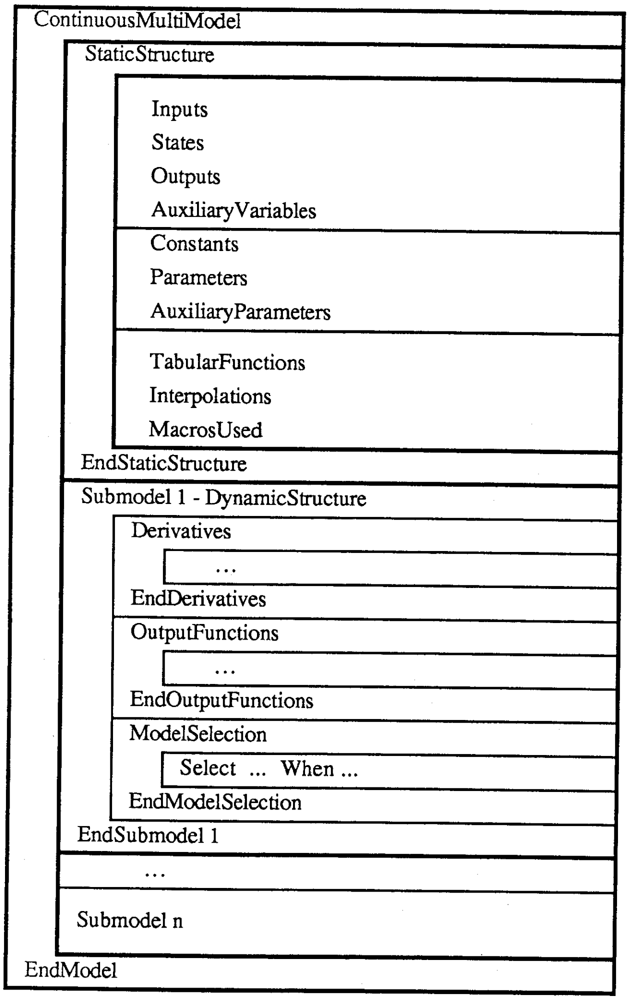

Figure 6. Continuous Multimodel

REFERENCES

Aytaç, K.Z. and T.I. Ören. 1986. "Magest: A Model-Based Advisor and Certifier for Gest Programs." In Modelling and Simulation Methodology in the Artificial Intelligence Era, M.S. Elzas, T.I. Ören, and B.P. Zeigler, eds. North Holland, Amsterdam, 299-307.

Birta, L.G., T.I. Ören, D.L. Kettenis. 1985. "A Robust Procedure for Discontinuity Handling in Continuous System Simulation." Transactions of the Society for Computer Simulation, 189-205.

Cellier, F.E. 1979. "Combined Continuous/Discrete System Simulation Languages - Usefulness, Experiences, and Future Development." In Methodology in Systems Modelling and Simulation, B.P. Zeigler et al., eds. North Holland, Amsterdam, 201-220.

Ören, T.I. 1971. Gest: A Combined Digital Simulation Language for Large-Scale Systems. In Proceedings of the Tokyo 1971 AICA Symposium on Simulation of Complex Systems (Tokyo, Japan, Sept. 3-7), Society of Analog Technique of Japan, Tokyo, Japan, B-1/1 - B-1/4.

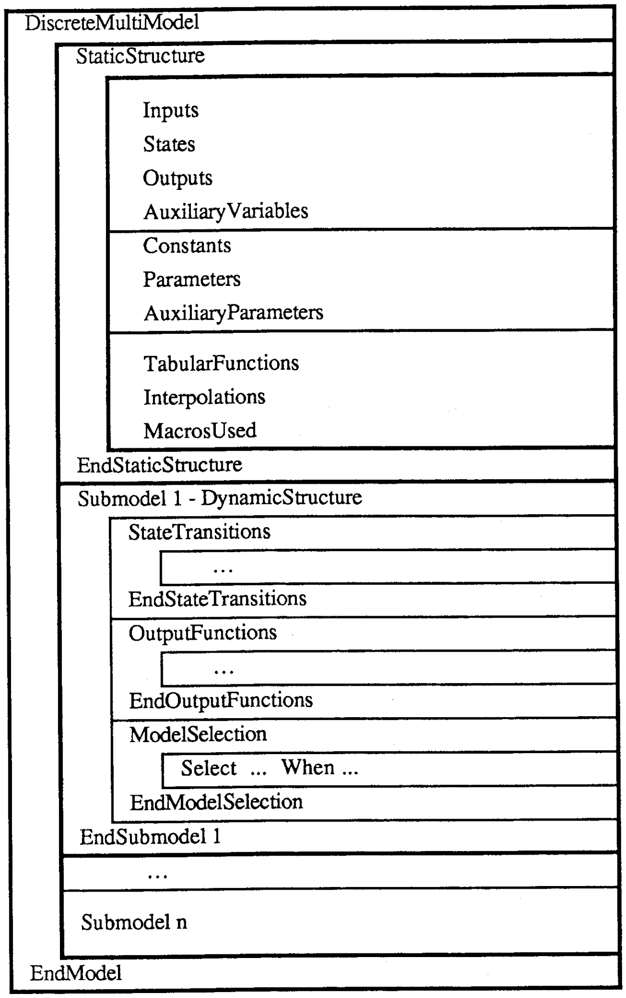

Figure 7. Discrete Multimodel

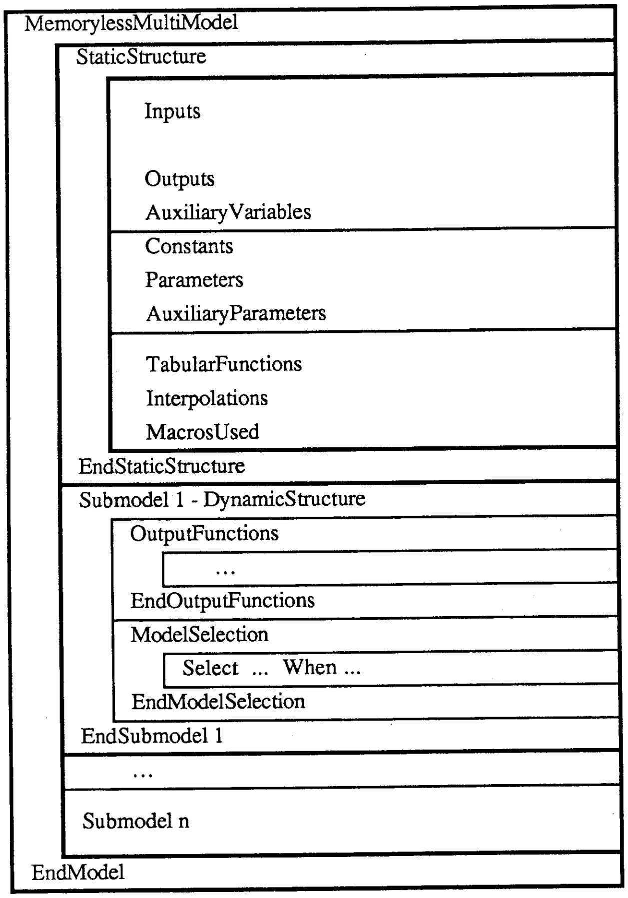

Figure 8. Memoryless Multimodel

Ören, T.I. 1984a. "Graphic Representation of Pseudocodes and Computer Programs: A Unifying Technique and a Family of Documentation Programs." In Proceedings of EdCompCon-83 (First Educational Computing Conf. of IEEE Computer Society) (San Jose, Calif., Oct. 18-20), D.C. Rine, eds, IEEE Computer Society, Piscataway, NJ, 81-89.

Ören, T.I. 1984b. "Gest - A Modelling and Simulation Language Based on System Theoretic Concepts." In Simulation and Modelling Methodologies: An Integrative view, T.I. Ören, B.P. Zeigler, M.S. Elzas, eds. Springer-Verlag, Heidelberg, Germany, 281-335.

Ören, T.I. 1987. "AI and Simulation: From Cognitive Simulation Toward Cognizant Simulation." Simulation, 48:4.

Ören, T.I. and Ma Jihu. 1986. "An Adaptive Order Discontinuity Algorithm for Simulation of Ordinary Differential Equations." In Proceedings of JSST (Japan Society for Simulation Technology) Conference on Recent Advances in Simulation of Complex Systems (Tokyo, Japan, July 15-17), 283-288.

Ören, T.I. and M. Özmızrak. 1987a. "Gest87 Definition." Technical Report, Computer Science Department, University of Ottawa, Ottawa, Ont., Canada.

Ören, T.I. and M. Özmızrak. 1987b. "Modelling and Simulation in Gest87: Tutorial." Technical Report, Computer Science Department, University of Ottawa, Ottawa, Ont., Canada.

Tuncer Ören
montréal
1987.07.26

PROCEEDINGS
OF THE
1987
SUMMER COMPUTER
SIMULATION CONFERENCE

JULY 27-30, 1987
THE QUEEN ELIZABETH HOTEL
MONTREAL, QUEBEC, CANADA

NINETEENTH ANNUAL SUMMER
COMPUTER SIMULATION CONFERENCE

Edited by
Jordan Q.B. Chou

CONFERENCE SPONSOR

SCS

THE SOCIETY FOR COMPUTER SIMULATION

Printed in the United States of America

GROUP XV: SIMULATION OF DISCRETE SYSTEMS

SESSION 1: Simulation of Discrete Systems

SESSION 2: Discrete Event Simulation Languages

GROUP XVI: FLIGHT SIMULATION SYSTEMS

SESSION 1: Motion Systems for Flight Simulators

SESSION 2: Aerodynamic Simulation

SESSION 3: Visual Systems for Flight Simulators

SESSION 4: PANEL: Flight Simulation Sickness

GROUP XVII: MILITARY AND DEFENSE SYSTEMS

SESSION 1: Model Design

SESSION 2: Simulation Tools