MODELING AND SIMULATION OF LARGE-SCALE HYBRID SYSTEMS

Manuel A. Pereira Remelhe and Sebastian Engell
Universiät Dortmund, Germany.

Keywords: Hybrid systems, large systems, object-oriented modeling, DAE-systems, discrete event formalisms, supervisory control, variable structure systems, state events

Content

Summary

In this chapter concepts and tools for modeling and simulation of large-scale multi-domain hybrid systems including complex discrete event controllers are presented. The object-oriented modeling paradigm is discussed in detail, because it is a very promising approach for modeling hybrid physical systems. The representation of a physical system is a set of differential and algebraic equations (DAE). Physical discontinuities can be considered via conditional and instantaneous equations. These require specific hybrid simulation features, such as state event detection and localization, event-iteration, re-initialization after discontinuities, etc. Purely discrete event supervisory controllers are best modeled using problem specific formalisms, such as automata, statecharts, logic diagrams, etc. It is explained how these can be integrated with object-oriented models of the physical systems.

1. Introduction

Sophisticated technological systems such as automatic gearboxes, robots, aircraft, and chemical plants consist of a large number of physical components, numerous low-level set-point controllers, interlocks, and interacting complex supervisory controllers. From a radical point of view, all observable behaviors of these systems are caused by quantum mechanics and hence discrete in nature. However, in most cases, engineers are interested only in the resulting macroscopic effects. Therefore, it is rational and common practice to apply the concepts of classical physics to model physical components. Set point controllers may be considered as continuous or discrete time systems depending on the sampling rates and the relevant time scale.

On the supervisory control level, fault detection, redundancy management, and sequence control (e.g., for start-up and shutdown) are performed, and the interaction with the user is managed. The dominant part of the functions on this level consists of logic operations that are triggered when continuous input signals (e.g., temperature measurements) cross specified thresholds, when discrete input signals change, notifying that certain events occurred in the environment of the controller (e.g., an input signal from a limit switch device), or when internal timers reach given time limits. The states and the outputs change discontinuously when a reaction to external stimuli is required. Supervisory controllers can often be regarded as ideal discrete event systems where the state changes instantaneously. For digital controllers it is often rational to ignore sampling effects so that their reactions also occur instantaneously.

Systems in which discrete event and continuous dynamics interact are called *hybrid systems*. The physical system part may consist of subsystems from various domains: electrical circuits, pneumatic and hydraulic actuators, mechanical transmissions, fuel cells, combustion chambers, tanks, gas transport systems, chemical reactors, etc. Each modeling domain has specific graphical representations and modeling traditions, but in most cases the final models are differential and algebraic equations (DAE) involving continuous variables that depend on continuous time.

The models of the physical components may contain discontinuities that strictly speaking are caused by model simplifications which are made in order to avoid models with largely different time scales. Examples are thermodynamic phase changes, friction and ideal switches, e.g. diodes in electronic systems or ideal valves. Other discontinuities occur when physical limits are reached (e.g. the overflow of a tank or contact in mechanical systems) or inputs to the physical system change abruptly. At points of discontinuity, even the number of independent state variables may change, e.g., if two rigid bodies make contact. In consequence, the physical part of the system itself may exhibit hybrid behavior, i.e. mixed discrete / continuous dynamics.

The modeling of such large-scale multi-domain hybrid systems including complex discrete event controllers can be very time consuming and expensive, but in many cases it is necessary for studying the overall performance of a system in the design stage.

For instance, in order to verify that an aircraft operates according to the requirements even when an actuator of the elevator system fails, a model has to be used that includes the general aircraft dynamics, the discrete event redundancy management system that normally consists of two independent controllers and detailed models of the actuator dynamics. Further simulation goals that require to model complete systems may be the estimation of throughput or power consumption, a feasibility check for a specific production plan, or operator training.

2. General Concepts

When modeling large-scale multi-domain hybrid systems, it is crucial to keep the modeling process as simple and as clear as possible in order to achieve a tolerable modeling effort and to reduce the error-proneness as far as possible. This section describes the main concepts and principles for powerful and efficient modeling:

- compositional modeling, modularity and hierarchy
- congruence of system and model with respect to structure and interfaces
- non-causal modeling of physical systems
- heterogeneous modeling and use of domain-specific formalisms

A well known means to cope with complexity is to decompose a problem into sub-problems and to synthesize the solution of the complex problem as a composition of basic sub-problem solutions. In compositional modeling the overall system is separated from its environment and decomposed into sub-components. If sub-components are further decomposed, a hierarchical model structure is obtained. The basic components are described more or less independently, resulting in basic component models that can be coupled in order to define the behavior of higher level components. In a modular model the components interact only via their interfaces, rendering all interactions apparent to the modeler and allows the reuse of component models in different settings without unexpected side effects.

In order to obtain an intuitive model, the model components should correspond to components of the real system. In particular the interfaces of the model components should be congruent with the interfaces of the real components so that the overall model can be composed analogously to the structure of the real system. Any departure from this principle unnecessarily renders the modeling more difficult and is a potential source of confusion.

For instance, in chemical batch plants the production is typically controlled by sequential controllers that implement the recipes. Usually, these controllers are software components within a more complex control system that may have concurrent parts and hierarchically structured components. Some of the recipe transitions are triggered by thresholds of continuous input signals that are evaluated *within* those software components so that the interface of those controllers should incorporate all continuous input signals needed. Unfortunately, many modeling formalisms and tools required to extract the thresholds from discrete models, to replace them by symbols or Boolean signals, and to evaluate the thresholds outside of the discrete event part, which results in a structural model-system-mismatch.

For the modeling of physical systems it is advantageous to apply a non-causal modeling approach, since physical laws are non-causal in nature: they define the potential behavior of physical systems without implying cause-and-effect-relationships. Consider for example Ohm's law of an electrical resistor, $\Delta u = R \cdot i$. This equation declares a quantitative relationship between the voltage drop Δu across the resistor and the current i through the resistor, but it does not tell whether the voltage drop causes the current or vice versa. In practice, non-causal modeling means that the interface variables of the model components are not predetermined as inputs or outputs, and that model equations are interpreted as equality constraints but not as computational statements. The advantage of this approach can be illustrated with the electrical resistor. When using a causal representation, one needs two different models of an electrical resistor, because it depends on the structure of the circuit whether the voltage drop has to be computed from the current, or vice versa. Hence, in one model the current would be the input variable and the voltage drop the output, whereas in the second model, the voltage drop is the input and the current is the output. If a change is made in the overall circuit model, the causality may change so that some component models have to be replaced. In a non-causal approach, the model equations are solved or ordered automatically to provide the needed *computational causality* when the overall model is prepared for simulation, so that only one model is needed for all possible configurations.

Many different graphical, textual and formal representations are used for specification in systems engineering. They may originate from computer science, control theory, other engineering disciplines, or the traditions of particular industrial sectors. Often even a specific combination of different formalisms is used to maintain transparency of the description, e.g. block diagrams combined with finite state machines. Each formalism or language has a specific syntax and semantics that is well suited for the particular application and which the users consider to be “natural”. In order to keep the modeling effort low and to provide the user with an intuitive model, the original specifications should be used for the modeling of the systems whenever possible. This requires that modeling environments support the use of domain-specific formalisms and heterogeneous modeling, i.e. to combine components that are specified by different modeling languages in one model.

3. System Representations and Software Tools

In order to give an impression of the variety of representations and tools, the most interesting of them are briefly presented in this section.

3.1. Representations of Discrete Event and Continuous Systems

In the specification and implementation of logic controllers, problem specific discrete event formalisms are usually employed. Popular formalisms are Automata, Statecharts, Petri Nets, Dataflow Diagrams, Synchronous Languages, or programming languages such as Sequential Function Charts and Function Block Diagrams as specified in the IEC 61131-3 standard for programmable logic controllers. An alternative to the use of domain-specific formalisms and languages would be a transformation to one general format, e.g. interacting automata. However, the transformation of discrete event models from one formalism into another is complicated and often leads to inefficient (i.e. unnecessarily large) models, even for formalisms with exactly or nearly equivalent expressive power.

Similarly, in the engineering of the physical part of a system, frequently a combination of problem specific representations is used. Many simulation tools on the market are tailored to the particular needs of a certain industrial sector and a specific application area (e.g. AMESIM for fluid power systems and hydraulics, SIMPACK for multi-body simulation, HYSYS for chemical plants, etc.). Generally, these tools are very convenient to use within their scope, because the modeling is based on predefined components that correspond to physical entities, so that the resulting model structure reflects the topology of the physical system. However, the perfect adaptation to one application area makes such tools very difficult to use for the modeling of large-scale multi-domain systems.

As an alternative general purpose modeling tools are available that support abstract formalisms such as block diagrams where each block is described by an ODE or an algebraic equation. Using these tools, the modeler has to transform the physical description of the system into a specific mathematical form. After this transformation, the topology of the physical system, e.g. the objects and the connections within an electrical circuit, is not explicitly visible in the model structure anymore. An intermediate position is taken up by bond graphs. They represent the flow of power through a system in an abstract way and maintain the topology of the physical system.

In contrast to the discrete event world, where several computational models are applied, ordinary differential equations (ODEs) and differential and algebraic equations (DAEs) represent a widely accepted general formalism for continuous systems. Virtually every continuous simulation tool uses standard or tailored ODE- or DAE-Solvers.

A very promising approach to multi-domain modeling of physical systems is the object-oriented modeling paradigm, because it adopts a modular and non-causal modeling style. This allows us to define domain-specific component libraries (additional to existing standard libraries) in a transparent way: The developer has to specify the non-causal interfaces, the internal variables and the equations of each component. This approach will be explained in section 4.

-
-
-

Bibliography

Alur, R., Dang, T., Esposito, J., Hur, Y., Ivančić, F., Kumar, V., Lee, I., Mishra, P., Pappas, G.J., Sokolosky, O. (2003). Hierarchical modeling and analysis of embedded systems. *Proceedings of the IEEE*, 91(1):11-28. [This paper describes the modeling language CHARON. It is shown how the modular structure of the language can be exploited by analysis tools and is supported by a formal semantics with an accompanying compositional semantics.]

Barton, P. I. (2002). Modeling, simulation, and sensitivity analysis of hybrid systems: mathematical foundation, numerical solutions, and software implementations. *Proceedings of the 11th IEEE International Symposium on Computer Aided Control System Design*, Anchorage, Alaska, pp. 117-122.

van Beek, D.A. and Rooda, J.E. (2000). Languages and applications in hybrid modeling and simulation: Positioning of Chi. *Control Engineering Practice*, 8(1):81-91. [Many existing modeling languages and tools are classified with respect to five categories. The Chi language is presented and two examples are used to illustrate the suitability of Chi.]

Branicky, M.S., Mattson S.E. (1997). Simulation of hybrid systems in Omola/OmSim. In Boullart et al. (eds.), *Computer Aided Control Systems Design*, Gent, Belgium: Pergamon.

Cellier, F.E., Elmqvist, H., and Otter M. (1996). Modeling from physical principles. In W.S. Levine, editor, *The Control Handbook*, CRC Press, Boca Raton, FL, pp. 99-107. [A rationale on physical systems and object-oriented modeling]

Deshpande, A., Göllü, A., and Varaiya, P. (1997). Shift: A formalism and a programming language for dynamic networks of hybrid automata. In Antsaklis, P., Kohn, W., Nerode, A., and Sastry, S., editors, *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*, pp. 113-133, Springer. [This paper presents the Shift programming language for specification and simulation of dynamic networks of hybrid automata.]

Eker, J., Janneck, J., Lee, E. A., Liu, J., Liu, X., Jozsef, L., Neuendorffer, S., Sachs, S., and Xiong, Y. (2003). Taming heterogeneity - the Ptolemy approach. *Proceedings of the IEEE*, 91(1):127-144. [This paper proposes a model structure and a semantic framework that supports hierarchical heterogeneity and discusses the issues arising from heterogeneous component interaction and the desire for component reuse. It introduces the notion of domain polymorphism as a way to address these issues.]

Henzinger, T.A. (1996) The Theory of Hybrid Automata. *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science* (LICS 1996), pp. 278-292. [The author classifies hybrid automata according to what questions about their behavior can be answered algorithmically. Certain classes of hybrid automata permit the application of various model-checking techniques that were originally developed for finite-state systems.]

IEEE Standards Association (1999). IEEE standard 1076.1-1999. [The VHDL-AMS standard]

Kofman, E., Lee, J., and Zeigler, B. P. (2001). DEVS representation of differential equation systems: Review of recent advances. *Proceedings of the 2001 European Simulation Symposium*. Marseille, France, Oct 2001. [In this paper, two recently introduced DEVS-based techniques for simulation of continuous systems represented by systems of differential equations are discussed: quantized systems and quantized state systems.]

Lynch, N., Segala, R., and Vaandrager, F. (2003). Hybrid I/O Automata. *Information and Computation*, 185(1):105-157. [A formal description of Hybrid Input Output Automata]

Mosterman, P. J., Otter, M., and Elmqvist, H. (1998). Modeling Petri-nets as local constraint equations for hybrid systems using Modelica. *Proceedings of the 1998 Summer Computer Simulation Conference (SCSC'98)*, Reno, USA, pp. 314-319.[A technique for representing discrete event models in an equation-based fashion is illustrated by realizing a library for petri nets in Modelica (see chapter 4.3).]

Mosterman, P.J. (1999). An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pp. 165-177, Springer. [Simulation packages are evaluated with respect to event handling, run-time equation processing, discontinuous state changes, event iteration, chattering, and comparing Dirac pulses]

Otter, M., Elmqvist, H., and Mattson, S. E. (1999). Hybrid modeling in Modelica based on the synchronous data flow principle. In Gonzalez, O., editor, *Proceedings of the IEEE International Symposium on Computer Aided Control System Design, CACSD'99*, pp. 151-157. [This paper presents the features of Modelica to model hybrid systems. Parameterized curve descriptions are applied to several hybrid examples.]

Otter, M., Engell, S., and Mosterman, P. J. (2000). Hybrid models of physical systems and discrete controllers. *at - Automatisierungstechnik*, 48(9):426-437. [In this article a rectifier circuit containing 4 ideal diodes is used to illustrate some consequences of parameterized curve descriptions. It is discussed how the resulting set of equations can be solved numerically. Moreover, ideas on the integration of discrete event systems are presented.]

Pereira Remelhe, M.A., Engell S., Otter M., Deparade, A., and Mosterman P.J. (2002): An environment for the integrated modelling of systems with complex continuous and discrete dynamics. In S. Engell, G. Frehse and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, volume 279 of

Lecture Notes in Control and Information Sciences, pp. 369-390, Springer. [This article introduces the DES/M approach.]

Zeigler, B.P., Praehofer, H., Kim, T.G. (2000). *Theory of Modeling and Simulation – Integrating Discrete Event and Continuous Complex Dynamic Systems*. Second Edition. New York: Academic Press. [This book provides a comprehensive framework for continuous and discrete event modeling and simulation. Topics: DEVS and its extensions, morphisms for model abstraction and simplification, quantized systems, parallel and distributed simulation of discrete event models, etc.]

Zeigler, B. P., Song, H. S., Kim, T. G., and Praehofer, H. (1995). DEVS framework for modelling, simulation, analysis, and design of hybrid. In Antsaklis, P., Nerode, A., Kohn, W., and Sastry, S., editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pp. 529-551, Springer. [Some features of the DEVS formalism and its extensions are reviewed and it is shown how to use DEVS to formulate and synthesize supervisory level controller.]

Homepages of modeling languages and tools

20-sim	http://www.20sim.com
CHARON	http://www.cis.upenn.edu/mobies/charon/
Chi	http://se.wpa.wtb.tue.nl/~vanbeek/
DEVS	http://www.acims.arizona.edu/
Dymola	http://www.dynasim.se/
gPROMS	http://www.psenterprise.com/
Matlab	http://www.mathworks.com/
Modelica	http://www.modelica.org/
Ptolemy	http://ptolemy.eecs.berkeley.edu/
Shift	http://www.path.berkeley.edu/shift/
VHDL-AMS	http://www.eda.org/vhdl-ams/

Biographical Sketches

Manuel A. Pereira Remelhe was born Oct. 6, 1971 in Vila Nova de Gaia (Portugal). In 1997 he received the Dipl.-Ing. degree in Chemical Engineering from Universität Dortmund, Germany. Since 1997, he has been working as PhD Student at the Process Control Group of the Department of Biochemical and Chemical Engineering at the Universität Dortmund. In this period, he was active in several cooperative scientific programs: the DFG focused research program "Analysis and synthesis of mixed continuous discrete systems", the DFG focused research program "Integration of software specification techniques for applications in engineering", the Graduiertenkolleg "*Modelling and model-based design of complex technological systems*" at Universität Dortmund, and the European Network of Excellence HYCON. Since 2006, he is with Bayer Technology Services, Leverkusen..

Sebastian Engell (IFAC Fellow, Member IEEE) received the Dipl.-Ing. degree in Electrical Engineering from the Ruhr-Universität Bochum, Germany, in 1978, and the Dr.-Ing. degree from Universität Duisburg, Germany in 1981. In 1987, Universität Duisburg granted him the *venia legendi* in Automatic Control.

From 1981 to 1984 and 1985-86 he was a senior researcher in the Automatic Control Group in the Mechanical Engineering Department at the University of Duisburg, 1984/85 he spent a year at McGill University, Montréal. 1986 he joined the Fraunhofer-Institut IITB, Karlsruhe, Germany where he led projects on industrial control and production scheduling. For his work, he received a Joseph von Fraunhofer-Prize in 1991. Since 1990 he is Professor of Process Control in the Department of Biochemical and Chemical Engineering, Universität Dortmund, Germany. 1996-1999 he served as the Chairman of the department, from 2002-2006 he was Vice-Rector for Research and International Relations of Universität Dortmund. He was Co-Editor of the IEEE Transactions on Control Systems Technology 1992-2000 and Associate Editor of the European Journal of Control 1995-2000. He currently is Associate Editor of the Journal of Process Control and of Mathematical and Computer Modelling of Dynamic Systems. In 2006, he was appointed as a Fellow of the International Federation of Automatic Control (IFAC).

His areas of research are control design, modeling and control of chemical processes, hybrid systems, and production scheduling in the process industries.