Electronic Communications of the EASST
Volume 10 (2008)

Proceedings of the
Seventh International Workshop on
Graph Transformation and Visual Modeling Techniques
(GT-VMT 2008)

Foundations of Modelling and Simulation of Complex Systems

Hans Vangheluwe

12 pages

Guest Editors: Claudia Ermel, Reiko Heckel, Juan de Lara
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Foundations of Modelling and Simulation of Complex Systems

Hans Vangheluwe

hv@cs.mcgill.ca
School of Computer Science
McGill University, Québec, Canada

Abstract: Modelling and simulation are becoming increasingly important enablers for the analysis and design of complex systems. In application domains such as automotive design, the notion of a "virtual experiment" is taken to the limit and complex designs are model-checked, simulated, and optimized extensively before a single realization is ever made. This "doing it right the first time" leads to tremendous cost savings and improved quality. Furthermore, with appropriate models, it is often possible to automatically synthesize (parts of) the system-to-be-built.

In this paper, the basic concepts of modelling and simulation are introduced. These concepts are based on general systems theory and start from the idea of a model as an abstract representation of knowledge about structure and behaviour of some system. The purpose is either analysis or design in a particular experimental context.

Typically, different formalisms are used such as Ordinary Differential Equations, Queueing Networks, and State Automata. It will be shown how these different formalisms all share a common structure and differ in the choice of time base, state space, and description of temporal evolution. This allows one to classify formalisms on the one hand and to find a common ground for implementing simulators on the other hand.

Keywords: modelling, simulation, systems theory, multi-formalism

1 Introduction

At a first glance, it is not easy to characterize modelling and simulation. Certainly, a variety of application domains such as fluid dynamics, energy systems, and logistics management make use of it in one form or another. As a paradigm, it is an intergrated way of representing problems and thinking about them as much as a solution method. Many compelling reasons exist to use simulation experiments as a substitute for real-world experiments. The latter may be too costly or dangerous to perform, or un-ethical. As simulations may be performed in simulated-time rather than in real-time, it becomes possible to evaluate many model variants rapidly. This allows one to drastically optimize system designs before physical realization. After optimization, parts of the model may be amenable to application synthesis. The problems solved with modelling and simulation cover the analysis and design of complex dynamical systems. Complexity can be due to a large number of components, but also to the heterogeneity of those components. In analysis, abstract models are built inductively from observations of a real system. In design, models deductively derived from a priori knowledge are used to build a system, satisfying certain design goals.

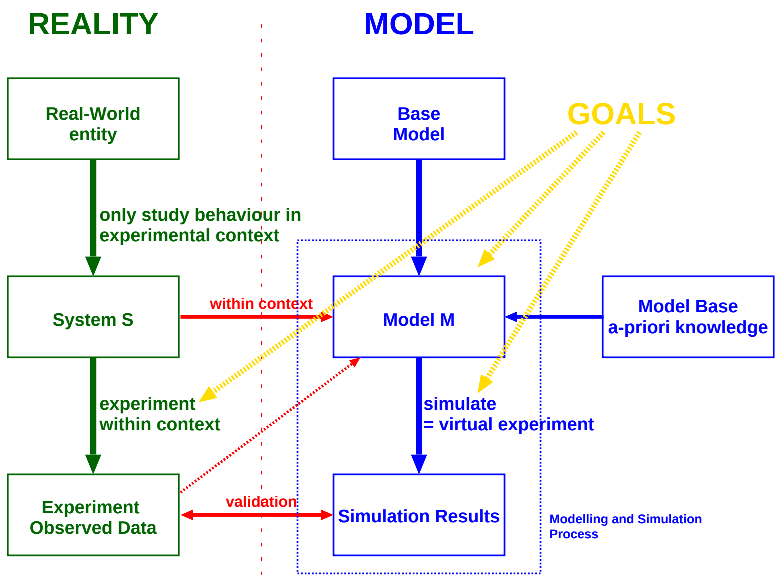
REALITY
MODEL
Real-World entity
Base Model
GOALS
only study behaviour in experimental context
System S
within context
Model M
Model Base a-priori knowledge
experiment within context
simulate = virtual experiment
Experiment Observed Data
validation
Simulation Results
Modelling and Simulation Process

Figure 1: Modelling and simulation concepts

Often, an iterative combination of analysis and design is needed to solve real problems. Though the focus of modelling and simulation is on the time-varying behaviour of dynamical systems, static systems are a limit-case. Both physical (obeying conservation and constraint laws) and non-physical (informational, such as software) systems and their interactions are studied.

This paper first introduces the general concepts of Modelling and Simulation theory. The representation of models in diverse formalisms, at different levels of abstraction, and the behaviour-conserving transformation between the formalisms is discussed. Multi-formalism model representation is presented as an enabler for open, distributed, modelling and simulation systems.

2 Modelling and Simulation Basics

2.1 Concepts

Figure 1 presents modelling and simulation concepts as introduced by Zeigler [ZPK00].

Object is some entity in the Real World. Such an object can exhibit widely varying behaviour depending on the context in which it is studied as well as the aspects of its behaviour which are under study.

Base Model is a hypothetical, abstract representation of the object's properties, in particular, its behaviour, which is valid in all possible contexts, and describes all the object's facets.

System is a well defined object in the Real World under specific conditions, only considering specific aspects of its behaviour.

Experimental Frame (EF) describes a limited set of circumstances under which a system (real of model) is to be observed or subjected to experimentation. As such, the Experimental Frame reflects the *objectives* of the experimenter who performs experiments on a real system or, through simulation, on a model.

(Lumped) Model (not to be confused with a lumped parameter model [Cel91]) is an abstract representation of a system within the context of a given Experimental Frame. Usually, certain properties of the system's structure and/or behaviour are reflected by the model (within a certain range of accuracy).

Experimentation is the act of carrying out an experiment. An experiment may interfere with system operation (influence its input and parameters) or it may not. As such, the experimentation environment may be seen as a system in its own right (which may itself be modelled in a lumped model). Also, experimentation involves observation. Observation yields *measurements*.

Simulation of a lumped model in a certain formalism (such as Petri Nets, Differential Algebraic Equations (DAE) or Bond Graphs) computes the dynamic input/output behaviour. Simulation may use symbolic as well as numerical techniques. Simulation, which mimics the real-world experiment, can be seen as *virtual experimentation*. Whereas the goal of modelling is to *meaningfully* describe a system, presenting information in an understandable, re-usable way, the aim of simulation is to be *fast and accurate*. Symbolic techniques are often favoured over numerical ones as they allow the generation of classes of solutions rather than just a single one (*e.g.*, $sin(x)$ as a solution to the harmonic equation as opposed to one single approximate trajectory solution). Furthermore, symbolic optimizations have a much larger impact than numerical ones thanks to their global nature. Crucial to the System–Experiment/Model–Virtual Experiment scheme is that there is a *homomorphic* relation between model and system: building a model of a real system and subsequently simulating its behaviour should yield the same results as performing a real experiment followed by observation and codifying the experimental results.

Verification is the process of checking the consistency of a simulation program with respect to the lumped model it is derived from.

Validation is the process of comparing experiment *measurements* with *simulation results* within the context of a certain Experimental Frame [Bal97]. When comparison shows differences, the formal model built may not correspond to the real system. A large number of matching *measurements* and *simulation results*, though increasing confidence, does *not prove* validity of the model however. For this reason, Popper has introduced the concept of *falsification* [Mag85], the enterprise of trying to disprove a model. It is to be noted that the correspondence in generated behaviour between a system and a model will only hold within the limited *context* of the Experimental Frame. Consequently, when using models to exchange information between agents, a model must always be matched with an Experimental Frame before use. Conversely, a model should never be developed without simultaneously developing its Experimental Frame. This requirement has its repercussions on the design of model representation languages.

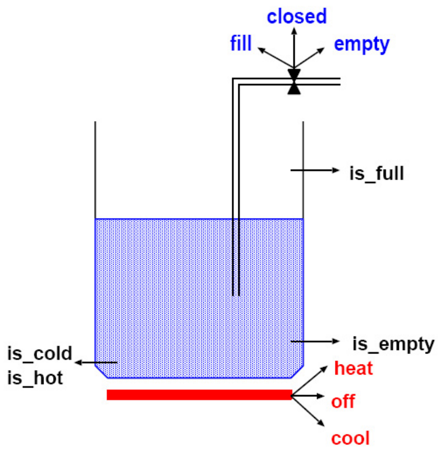

Figure 2: T,l controlled liquid

2.2 Abstraction and Formalisms

Abstract models of system behaviour can be described at different levels of abstraction or detail as well as by means of different formalisms. The particular formalism and level of abstraction depends on the background and goals of the modeller as much as on the system modelled. As an example, a temperature and level controlled liquid in a pot is considered. This is a simplified version of the system described in [BZF98], where structural change is the main issue. On the one hand, the liquid can be heated or cooled. On the other hand, liquid can be added or removed. In this simple example phase changes are not considered. The system behaviour is completely described by the following (hybrid) Ordinary Differential Equation (ODE) model:

$$\begin{cases} \frac{dT}{dt} = \frac{1}{l}[\frac{W}{c\rho A} - \phi T] \\ \frac{dl}{dt} = \phi \text{ if } 0 < l < H \text{ else } 0 \\ is\_low = (l < l_{low}) \\ is\_high = (l > l_{high}) \\ is\_cold = (T < T_{cold}) \\ is\_hot = (T > T_{hot}) \end{cases}$$

The inputs are the filling (or emptying if negative) flow rate ϕ, and the rate W at which heat is added (or removed if negative). This system is parametrized by A, the cross-section* surface of the vessel, H, its height, c, the specific heat of the liquid c, and ρ, the density of the liquid. The state of the system is characterized by variables T, the temperature and l, the level of the liquid. The system is observed through threshold output sensors $is\_low, is\_high, is\_cold, is\_hot$. Given input signals, parameters, and a physically meaningful initial condition (T_0, l_0), simulation of the behaviour yields a continuous state trajectory. By means of the binary (on/off) level and temperature sensors introduced in the differential equation model, the state-space may be discretized. The inputs can be abstracted to heater heat/cool/off and pump fill/empty/closed. At this level of abstraction, a Finite State Automaton (with 9 possible states) representation of the

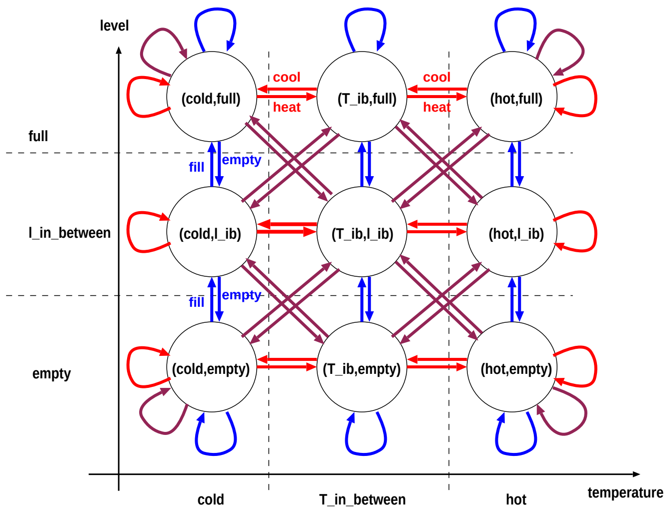

Figure 3: FSA formalism

dynamics of the system as depicted in Figure 3 is most appropriate.

Though at a much higher level of abstraction, this model is still able to capture the essence of the system's behaviour. In particular, there is a *behaviour morphism* between both models: model discretization (from ODE to FSA) followed by simulation must yields the same result as simulation of the ODE followed by discretization.

2.3 Systems Theory

In general systems theory [Wym67], a causal (output is the consequence of input), deterministic (input will lead to unique output) system model SYS is defined. It is a template for a plethora of different formalims such as Ordinary Differential Equations, Finite State Automata, Difference Equations, *etc.*Its general form is

$$SYS \equiv \langle T, X, \Omega, Q, \delta, Y, \lambda \rangle$$

T	time base
X	input set
$\Omega = \{\omega : T \to X\}$	input segment set
Q	state set
$\delta : \Omega \times Q \to Q$	transition function
Y	output set
$\lambda : Q \to Y$ (or $Q \times X \to Y$)	output function

$$\forall \omega, \omega' \in \Omega, \delta(\omega \bullet \omega', q_i) = \delta(\omega', \delta(\omega, q_i)).$$

The time base T is the formalisation of the independent variable time. Different ordering relations (partial/total) can be used over T. Common time bases (with appropriate $<$ and $+$) are

- $T = \{NOW\}$. Models such as algebraic models are *instantaneous*. The time base is a singleton.

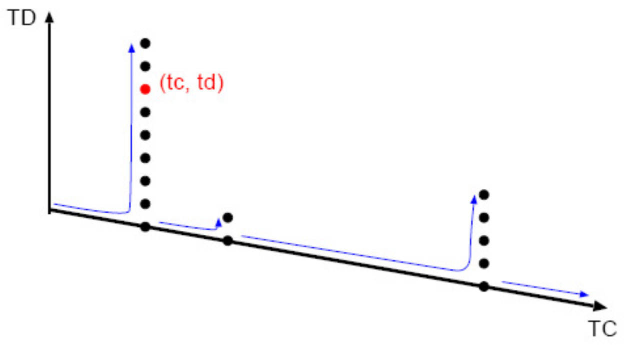

Figure 4: Time base for hybrid system models

- $T = \mathbb{R}$. Models with this time base are called *continuous-time* models. Note how *discrete event models* have $\mathbb{R}$ as a time base. However, only at a finite number of time-instants in a bounded time-interval, an event different from the non-event ϕ occurs.
- $T = \mathbb{N}$ (or isomorphic). Models with this time base are called *discrete-time* models. Some formalisms such as Finite State Automata (FSA) do not have an explicit notion of time (unlike their extension, timed automata). Hence, they are often called *untimed* models. There is however a notion of progression (from one state to another). According to our general definition, the index of progression, a natural number, is time.

In *hybrid* system models which combine aspects of continuous and discrete models [MB02], a system evolves continuously over time ($\mathbb{R}$) until a certain condition is met. Then, *instantaneously* (the continuous time does not progress), the system may go through a number of discrete states (the index of progression is discrete) before continuing its continuous behaviour. To uniquely describe progression (of generalized time) in this case, a tuple (t_c, t_d) depicted in Figure 4 is needed. Even when a series of discrete transitions keeps returning to the *same* state, the discrete index t_c allows one to distinguish between them. The time base used is

$$T = \{(t_c, t_d) | t_c \in \mathbb{R}, t_d \in \{1, \ldots, N(t_c)\}\}.$$

Here, $N(t_c)$ (≥ 1) describes the number of discrete transitions the system goes through at continuous time t_c. Obviously, only a partial ordering will be defined over T which consists of first testing the relationship between the t_c components, and subsequently (if equal), that between the t_d components.

In case of Partial Differential Equations (PDEs), the time base remains $\mathbb{R}$. The other *independent variables* (often space in the form of some coordinate system) should be seen as infinitely many state-variable *labels* or *generalized coordinates*.

The input set X describes all possible allowed input values (possibly a product set). An input segment ω represents input during a time-interval. The history of system behaviour is condensed into a *state* (from a state set Q). The dynamics is described in a transition function δ which takes a current state, and applies an input segment $\omega \in \Omega$ to it to obtain a new state. The system may generate output. This output is obtained as a function λ of the state (and more generally, of

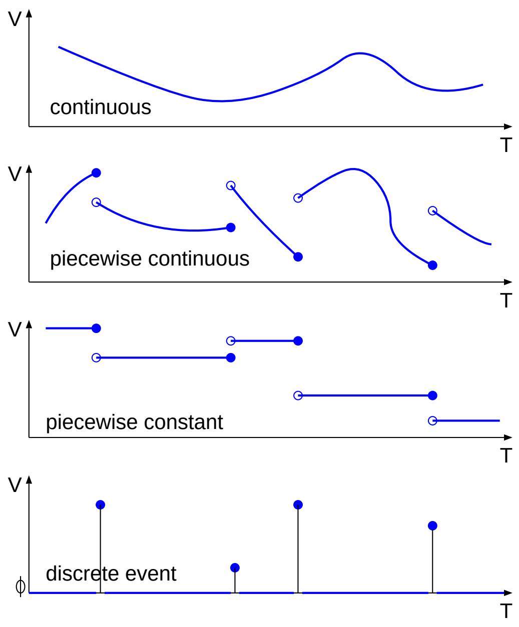

Figure 5: Segment types

the current input too). State and transition function must obey the composition or semigroup property. This property, whereby a transition over a time interval $[t_i, t_f]$ can always be split into a composition of transitions over arbitrary sub-intervals, is the basis of all model simulators.

Figure 5 shows some common segment types: continuous, piecewise continuous, piecewise constant and discrete event.

Note how for discrete event systems, inputs and output segments are *event segments*

$$\omega : \langle t_1, t_2 \rangle \rightarrow A \cup \{\phi\},$$

with ϕ the non-event. For such systems, the internal state behaviour is piecewise constant (the internal state only changes at event times).

A host of formalisms are currently in use. In particular, through time-scale or parameter abstraction, reality is often represented by means of *discrete-event models*. In these models, time evolves continuously ($T = \mathbb{R}$), but the state of the system only changes at a finite number of point in time in a bounded time-interval. These times are called event-times. A number of discrete-event formalisms, called *world views* were constructed. World views range from Event Scheduling which expresses events and their consequences explicitly, focussing on the simulation efficiency, over Activity Scanning and the Three Phase Approach which emphasize changing conditions in the system, to Process Interaction which represents the system in terms of interacting processes [Bal88, CS92]. Zeigler [Zei84] developed the DEVS discrete-event system specification. It is primitive in that it allows other discrete-event formalisms to be expressed in terms of DEVS. It is modular as it allows compositional construction of hierarchical models.

	T: **Continuous**	T: **Discrete**	T: {**NOW**}
Q: **Continuous**	DAE	Difference Equations	Algebraic Equations
Q: **Discrete**	Discrete event	Finite State Automata	Integer Equations
	Naive Physics	Petri Nets	

Table 1: I/O system model classification

2.4 System classification

Formalisms can be classified based on the general system model structure (in particular, the type of time base T and state set Q) as shown in Table 1. More specific classifications are based on the **structure** of formalisms. For continuous models, further classification according to physical domains such as mechanical, electrical, and hydraulical, is meaningful. The variety of classifications leads to the insight that ultimately, one should picture a vast formalism-space and classify that space according to various criteria. Different criteria will lead to different equivalence classes. Note how several exisiting modelling languages and tools may implement a single formalism.

3 Complex systems

3.1 Multi-component specifications

A common means to tackle complexity is to decompose a problem *top-down* into smaller sub-problems. Conversely, complex solutions may be built *bottom-up* by combining primitive sub-problem solution building blocks. Both approaches are instances of *compositional modelling*: the composition of *interacting component* models to build new models. In case the components *only* interact via their interfaces, and do not influence each other's internal working in any other way (model information is completely *encapsulated* in object-oriented terminology, the compositional modelling approach is called *modular*. If inter-components access is not restricted to take place via interfaces or ports only, the approach is called *non-modular*.

3.2 Multi-formalism modelling

Complex systems are characterized, not only by a large number of components, but also by the diversity of the components. For the analysis and design of such complex systems, it is no longer sufficient to study the diverse components in isolation, using the specific formalisms these components were modelled in. Rather, it is necessary to answer questions about properties (most notably behaviour) of the whole system. To focus the attention, Figure 6 gives an example of a complex system.

The complexity lies in the diversity of the different components, both in abstraction level and in formalism used:

- A paper and pulp mill produces paper from trees with polluted water as a side-effect. This

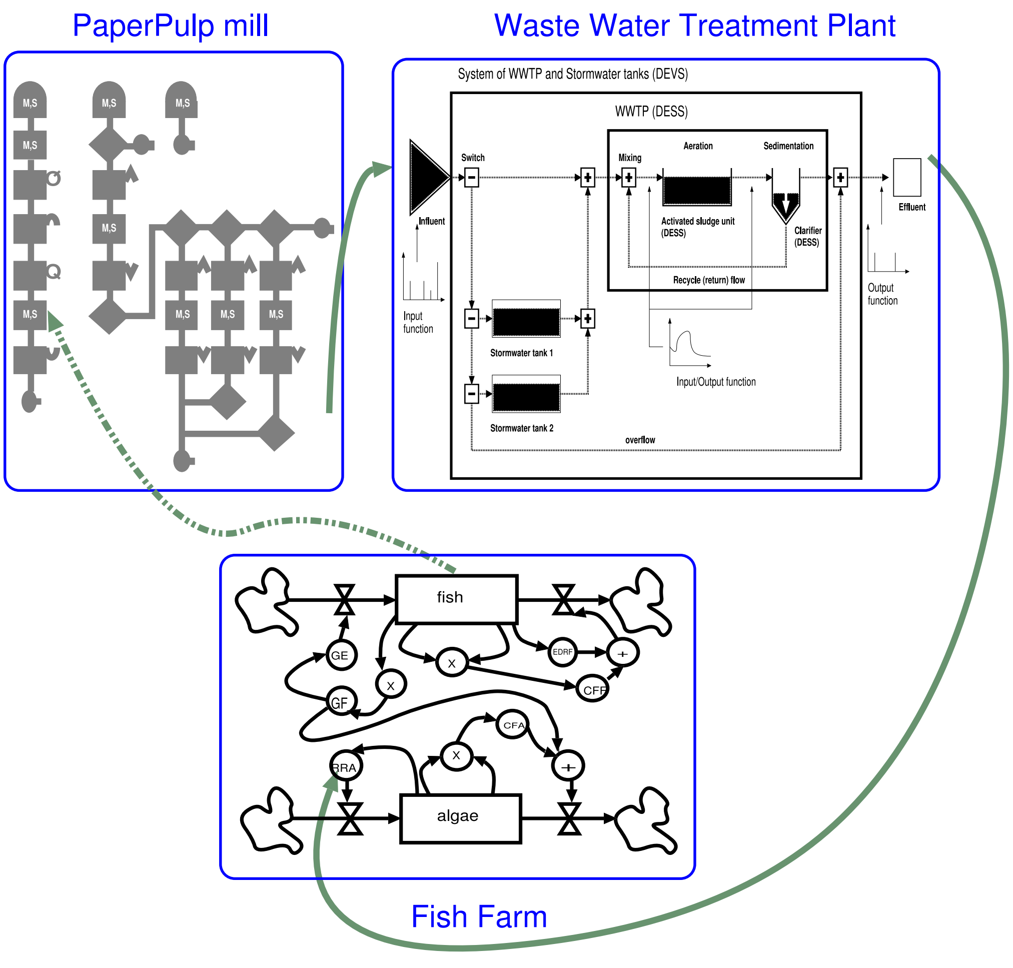

Figure 6: Complex system example

system is modelled as a process interaction discrete-event scheduling system.

- A Waste Water Treatment Plant (WWTP) takes the polluted effluent from the mill and purifies it. Some solid waste is taken to a landfill whereas the partially purified water flows into a lake. This system is modelled using Differential Algebraic Equations (DAEs) describing the biochemical reactions in the WWTP.
- A Fish Farm grows fish in the lake. Fish feed on algae which are highly sensitive to polluted water. The water is also used for a tree plantation which supplies the paper mill. This system is modelled using the System Dynamics formalism. The dotted feedback arrow from the fish farm to the paper mill indicates the possible disastrous impact of poisoned fish on the productivity of workers in the mill.

It is obvious that decision-making for this system will require understanding of the behaviour of the overall system. Studying the individual components will not suffice. The complexity of this system and its model is due to

- the number of interacting, coupled, concurrent components. Complex behaviour is often a consequence of a large number of feedback loops.
- the variety of components such as software/hardware, continuous/discrete.
- the variety of views, at different levels of abstraction.

A model of a system such as the one described above may be valid (within a particular experimental context) at a certain *level of abstraction*. This level of abstraction, which may be different for each of the components, is determined by the available knowledge, the questions to be answered about the system's behaviour, the required accuracy of answers*etc.*Orthogonal to the choice of model abstraction level is the selection of a suitable *formalism* in which the model is described. The choice of formalism is related to the abstraction level, the amount of data that can be obtained to calibrate the model, the availability of solvers/simulators for that formalism as well as to the kind of questions which need to be answered.

Milner, in his Turing Award lecture [Mil93] rejects the idea that there can be a unique conceptual model, or one preferred formalism, for all aspects of something as large as concurrent systems modelling. Rather, many different levels of explanation, different theories, languages are needed. We believe this view is amplified when arbitrarily complex systems are studied.

As an introduction to the semantics of multi-formalism models we present the Formalism Transformation Graph (FTG) in Figure 7.

The different formalisms are shown as nodes in the graph. The horizontal line at the bottom denotes the level at which trajectories can be described. The vertical dashed line demarcates continous model formalisms (on the left) from discrete model formalisms (on the right). The FTG shows a plethora of formalisms, indicating that in general, many classifications are possible. It suffices to annotate the nodes in the FTG with attributes (possibly derived from the formalism structure) and determine equivalence classes based on those attributes. The arrows denote a homomorphic relationship "can be mapped onto", implemented as a symbolic property-preserving transformation between formalisms. The vertical, dotted arrows denote the existence of a *solver* or *simulation kernel* which is capable of simulating a model, thus generating a trajectory. A trajectory is really a model of the system in the data formalism (time/value tuples). Obviously, the experimental frame of a trajectory is very limited. In a denotational sense traversing the graph makes *semantics* of models in formalisms explicit: the meaning of a model/formalism is given by mapping it onto some known formalism (whose meaning may in turn be given by mapping it onto an even more basic formalism*etc.*). Though a multi-step mapping may seem cumbersome, it can be perfectly and correctly performed by tools. The advantage of this approach is that the introduction of a *new formalism* only requires the description of the mapping onto the nearest formalism as well as the implementation of a translator to that formalism. It is often meaningful to introduce a new, domain-specific formalism for a specific application, encoding particular properties and constraints of the application. Often, translation involves some loss of information, though behaviour must obviously be conserved. This loss may be a blessing in disguise as it entails a reduction in complexity, hopefully leading to an increase in (simulation) performance. Usually, the aim of multi-step mapping is to eventually reach the trajectory level. Another major use for formalism transformation is the answering of particular questions about the system. Some questions can only be answered in the context of a particular formalism.

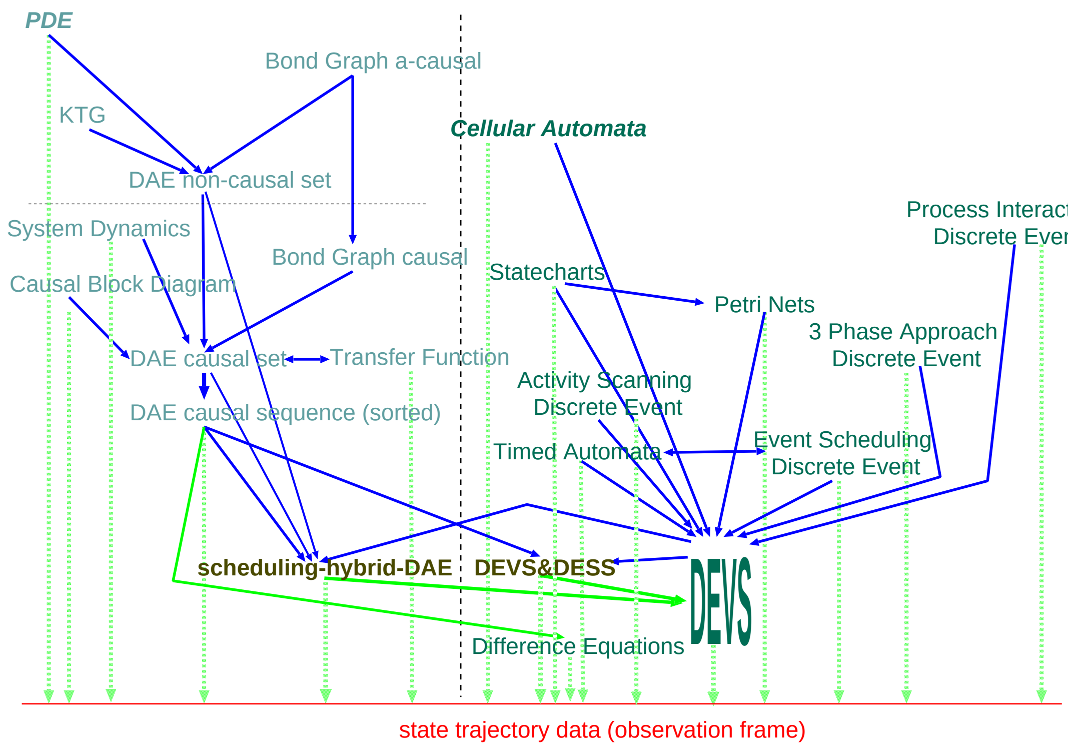

Figure 7: The Formalism Transformation Graph (FTG)

Note that in the *co-simulation approach*, each of the sub-models in a coupled model is simulated with a formalism-specific simulator. Interaction due to coupling is resolved at the trajectory level. Compared to transformation to a common formalim before simulation, this approach, though appealing from a software engineering point of view (it's object-oriented) discards a lot of useful information. Questions can *only* be answered at the trajectory level. Furthermore, there are obvious speed and numerical accuracy problems for continuous formalisms [FY97]. The approach is meaningful only for discrete-event formalisms. In this realm, it is the basis of the DoD High Level Architecture (HLA) for simulator interoperability.

4 Conclusions

In this paper, the basic concepts of modelling and simulation were introduced. These concepts are based on general systems theory and start from the idea of a model as an abstract representation of knowledge about structure and behaviour of some system. This lead to the concept of multi-formalism modelling of complex systems, and its relation to the semantics of models.

Bibliography

[Bal88] O. Balci. The implementation of four conceptual frameworks for simulation modeling in high-level languages. In Abrams et al. (eds.), *Proceedings of the 1988 Winter Simulation Conference*. Pp. 287–295. Society for Computer Simulation International (SCS), 1988.

[Bal97] O. Balci. Principles of Simulation Model Validation, Verification, and Testing. *Transactions of the Society for Computer Simulation International* 14(1):3–12, March 1997. Special Issue: Principles of Simulation.

[BZF98] F. J. Barros, B. P. Zeigler, P. A. Fishwick. Multimodels and Dynamic Structure Models: an Integration of DSDE/DEVS and OOPM. In Medeiros et al. (eds.), *Proceedings of the 1998 Winter Simulation Conference*. Pp. 413–419. Society for Computer Simulation International (SCS), 1998.

[Cel91] F. E. Cellier. *Continuous System Modeling*. Springer-Verlag, New York, 1991.

[CS92] B. A. Cota, R. G. Sargent. A Modification of the Process Interaction World View. *ACM Transactions on Modeling and Computer Simulation* 2(2):109–129, April 1992.

[FY97] L. Foster, K. Yelmgren. Accuracy in DoD High Level Architecture Federations. In Obaidat and Illgen (eds.), *Summer Computer Simulation Conference (SCSC'97)*. Pp. 451–460. Society for Computer Simulation International (SCS), July 1997. Arlington, Virginia.

[Mag85] B. Magee. *Popper*. Fontana Press (An Imprint of HarperCollins Publishers), London, 1985.

[MB02] P. J. Mosterman, G. Biswas. A Modeling and Simulation Methodology for Hybrid Dynamic Physical Systems. *Transactions of the Society for Computer Simulation International*, 2002.

[Mil93] R. Milner. Elements of Interaction. 36(1):70–89, January 1993. Turing Award Lecture.

[Wym67] A. W. Wymore. *A Mathematical Theory of Systems Engineering – the Elements*. Wiley Series on Systems Engineering and Analysis. Wiley, 1967.

[Zei84] B. P. Zeigler. *Multifacetted Modelling and Discrete Event Simulation*. Academic Press, London, 1984.

[ZPK00] B. P. Zeigler, H. Praehofer, T. G. Kim. *Theory of Modelling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, second edition, 2000.