Chapter 1
Introduction

1.1 A Historical Survey of Bond Graph Modelling

When designing a new dynamic system or analysing an existing one, it is common for designers and engineers to use graphical representations of their models in order to communicate with others, to exchange ideas, to express modelling assumptions and to exchange their models for reuse. This is not surprising, as graphical representations are far more suited to human perception than oral or textual ones. For instance, a schematic of a closed loop controlled system is clearly more easily understood than a verbal description given over the telephone. In engineering disciplines, e.g., network representations, block diagrams, or linear graphs have a long tradition. In addition, (domain specific) iconic diagrams have become popular. If graphical representations adhere to formal rules, not only do they avoid misunderstandings between human beings, but they also allow for an automatic transformation into an executable program by means of appropriate software programs.

Among several graphical representation means used in different application areas, bond graphs are a description formalism best suited for modelling physical processes and multidisciplinary dynamic engineering systems including effects or components from different energy domains, viz., the mechanical, the electrical, the thermal, and the hydraulic domain. Many technical systems, often termed *mechatronic* systems, integrate components from different disciplines and exploit interacting effects, e.g., sensors and electronically controlled actuators.

Bond graphs, to be formally introduced in the next chapter, were devised by Professor Henry Paynter[1] at the Massachusetts Institute of Technology (MIT) in 1959. His former Ph.D. students, Professor D. Karnopp and Professor D. Margolis (University of California at Davis), and Professor R. Rosenberg (Michigan State University, East Lansing, Michigan), elaborated the concept into a methodology for physical systems modelling that nowadays is used in academia and industry by many people around the world. In retrospect, Paynter wrote in 1992 ([35] p. 13):

[1] 1923–2002

So it was that on April 24, 1959, as the writer was about to give a seminar lecture at Case Institute (now Case-Western) on "Interconnected Engineering Systems", he awakened earlier that morning with the 0,1-junctions somehow finally planted in his head! Thus on that date the BG[2] *system was complete and constituted a formal discipline.*

The first published books of these pioneers include Paynter's historic lecture notes titled *Analysis and Design of Engineering Systems*, dating back to the year 1961 [34], the book titled *Analysis and Simulation of Multiport Systems – The Bond Graph Approach to Physical System Dynamics* by Karnopp and Rosenberg [22] published in 1968, as well as the first edition of the textbook *System Dynamics – A Unified Approach* [23]. This book has become a widely recognised standard. A second edition was published in 1990 and a third edition in 2000, both co-authored by D. Margolis. In 2006, the three authors published an even more mature fourth edition that reflects their experience over decades in teaching courses at universities and in industry. This textbook now is titled *System Dynamics - Modeling and Simulation of Mechatronic Systems* [25].

Early promoters of the new modelling technique were Professor J. Thoma (Professor Emeritus at the University of Waterloo, Ontario, Canada), Professor J. J. van Dixhoorn (University of Twente, Enschede, Netherlands who passed away in 2001), P. Dransfield, Professor at Monash University, Melbourne, Australia and S. Scavarda, Professor at Institut National des Sciences Appliquées de Lyon (INSA), France who passed away in 2008. These gentlemen significantly contributed to the promotion and dissemination of the bond graph modelling technique in Europe, Japan, India and China. Van Dixhoorn founded a Technical Committee on Bond Graph Modelling (TC 16) as part of the International Association for Mathematics and Computers in Simulation (IMACS) that was chaired by J. Thoma for many years.

Right from its beginning, the bond graph methodology was supported by the famous ENPORT™ simulation program developed by R. Rosenberg. Its probably best known version has been ENPORT-4™. At that time, bond graphs were entered in alpha-numerical form in a so-called line code. Entries in that line code separated by commas denoted the type of a bond graph vertex followed by the numbers of the edges attached to that node. In his Ph.D. thesis [20], Professor J. Granda (California State University at Sacramento) developed a bond graph preprocessor that transformed the line code into equations for input into widely used simulation programs such as ACSL™[3] . In a further obvious step, he replaced the line code entry by a graphical editor.

Nowadays, several facts demonstrate the worldwide acceptance and the success of bond graph modelling methodology. During the first years of bond graph methodology development, almost everyone concerned with the new technique knew almost all publications and moreover, knew many members of the small but worldwide community of bond graph modellers personally. Today, the number of bond

[2] BG means Bond Graph

[3] ACSL, acslX, and PowerBlock are registered trademarks of The AEgis Technologies Group, Inc., 631 Discovery Drive, Huntsville, AL 35806, USA, http://www.acslx.com

graph related publications has grown tremendously so that a comprehensive survey is almost impossible. In 1977, V. Gebben [19] published the first *Bond Graph Bibliography* with the aim of recording the enormous increase in the number of bond graph related publications. Updates followed in 1985 [6] and 1991 [8], both published in the renowned *Journal of the Franklin Institute*. Around 1996, Professor F. Cellier, now with the Swiss Federal Institute of Technology, Switzerland, took on the tremendous burden of compiling references to bond graph related publications in a Bond Graph Compendium available on the World Wide Web [12]. Even this comprehensive compendium needs another update. For instance, in the year 2000 alone, several textbooks on bond graph modelling were published [3, 13, 24, 30, 41]. Some more recently published books on bond graph modelling are [25, 31, 37, 43].

In the past decades, bond graph researchers contributed many special sessions on bond graph modelling to international conferences. The author organised such special sessions as part of the ESM 1993, Lyon, France, of the 1994 and the 2003 Mathmod conference in Vienna, Austria, as part of the ESS 1997 in Passau, Germany and contributed to bond graph sessions organised by other members of the community, e.g., to CESA 1996 in Lille, France, and the CIFA 2008 in Bucarest, Romania. As general chairman, the author of this book organised the 2006 European Conference on Modelling and Simulation (ECMS 2006) held near Bonn, Germany [5]. This conference featured a well received track with three sessions devoted to bond graph modelling. Professor Cellier delivered a keynote speech by addressing his current research activities in bond graph modelling. During the last decade, many more conferences with papers, sessions, or even tracks addressing bond graph modelling took place. Space in this introduction does not allow for all of them to be listed.

Besides publications in international conferences, bond graph researchers have contributed to special journal issues, e.g., the 1999 special issue of *Simulation Practice and Theory* edited by J. U. Thoma and H. J. Halin [42], the 2002 special issue of the *Proceedings of the Institute of Mechanical Engineers* edited by P. Gawthrop and S. Scavarda [17], the 2006 special issue of the journal *Mathematical & Computer Modelling of Dynamical Systems* edited by I. Troch, W. Borutzky and P. Gawthrop [44], or the 2009 special issue of the journal *Simulation Modelling Practice and Theory* edited by the author [4]. The latter special issue also includes an introduction of bond graph modelling by the author of this book. In 2007, Gawthrop and Bevan published a tutorial introduction into bond graph modelling for control engineers in the IEEE Control Systems Magazine [16].

Last but not least, the biannual *International Conference on Bond Graph Modelling* (ICBGM) as part of the Western Multiconference (WMC) of the Society for Modelling and Simulation International (SCS) has to be mentioned.

In addition to research, bond graph based physical systems modelling has been considered more and more to be a important fundamental topic in engineering education and has become a regular part in the syllabi of many engineering programmes. In addition to the 2000 textbooks already mentioned above, other textbooks on bond graph modelling have been published in different languages during the past decades, e.g. [2, 7, 14, 18, 23, 32, 40, 46]. This list, which is not meant to be comprehensive, shows that there has been a growing awareness and acceptance of the bond graph modelling technique worldwide during the past decades. In his contribution to the March, 1995 special issue on simulation in engineering of the journal SIMULATION, Professor F. Cellier points out why bond graphs are "the right choice for educating students in modelling continuous-time physical systems" [11].

Not only in research and education, but also in industry, bond graph modelling has become a useful approach in the everyday business of many engineers in small consulting firms as well as in big companies, especially the automobile industry, in aerospace and in automation. In the context of mechatronic system design, the appropriateness of a bond graph approach is particularly evident. On the other hand, it must be mentioned that bond graph modelling is one of a number of modelling methodologies that are equally well suited for a given design task. People in academia and industry do have their preferences. Consequently, bond graph modelling is not appreciated to the same extent in different places. Although bond graph modelling has spread from MIT to many places all over the world, there are engineering departments even in the USA where this technique is still not used.

In some countries, bond graph modellers founded national associations, e.g., the *Bondgraafclub* in the Netherlands, or the *Club de Bondgraphistes* with members in France and in Belgium.

In 2000, the first edition of this monograph was published as a first comprehensive presentation of bond graph methodology in the German language. The author of this monograph became aware of bond graphs in 1979 by a survey article by A. Schöne [38]. It was the starting point of an ever-continuing enthusiasm for this methodology that I share with friends in many places around the world.

1.2 Some General Aspects of Modelling Dynamic Systems

Building a model is an iterative procedure. It starts with the identification of essential features and of inherent mechanisms of a dynamic system to be designed. In a step by step refinement of the understanding of a dynamic physical system, different forms of representation are used. They are of graphical nature, especially during the conceptual phase as our eyes can easily perceive different information in parallel. Graphical representations are not only easy to grasp, they are also best suited for communication between individuals. This becomes evident if one considers the difficulty of transmitting all the information contained in a schematic of an electronic or hydraulic circuit by telephone, or if one receives the description of a dynamic system in a simulation language on many pages without any comments. The importance of graphical representations is not only essential in the modelling phase, but also for visualisation of simulation results. With the ever increasing computing power, not only is the graphical representation of numerical results as curves required, there is also a demand for 3D animation of system motion in a realistic fashion. With the development of languages such as Virtual Reality Modelling Language (VRML) and appropriate freely available visualisation software tools, 3D presentations may be exchanged via the internet. Thus, graphical representations and visualisation play a vital role during the design of a dynamic system and the design of its control [36].

Graphical and textual model representations of dynamic systems are always associated with a certain view of a system, its properties and its inherent mechanisms. They reflect abstractions and modelling assumptions. The starting point of the modelling process always involve certain questions. Thus, features and system properties are assessed and are either taken into account or deliberately neglected. These considerations and decisions as well as the designer's experience have an impact on the choice of the graphical description. Answers to the question as to what purpose a model of a system to be designed shall serve, may give an indication to an appropriate choice of a description formalism from a repertoire of possible means. Often, the answer to the question concerning the most suitable form of representation is not unique. Several description formalisms may equally serve the requirements, or depending on the design task, a combination of some of them may be appropriate, e.g., a bond graph representation of a system and a block diagram of its control. Moreover, a graphical representation may be transformed into another one provided both are equivalent. For instance, a network could be transformed into a bond graph which in turn could be transformed into a block diagram. Consequently, a program for control systems could be used to simulate a system described as a network. There is a similar situation in computer science. For the software implementation of a solution to a given problem, several programming languages may be appropriate, whereby each one has its own flavour. On the other hand, software programs for automatic translation from one programming language to another are available.

1.3 Object-Oriented Physical Systems Modelling

The ever increasing performance of computers and simulation software has enabled one to model and to simulate problems of more and more complexity. While the simulation of a problem required hours of computation time in former times on computers called minicomputers, the same problem can be solved within minutes or less on a personal computer. The ever increasing complexity of problems to be analysed has had an essential impact on modelling methodologies and the software tools supporting them.

In order to cope with the increasing complexity of the system to be analysed, it is obviously necessary to pursue a hierarchical approach and to build libraries with reusable submodels as is known, for example, from the design of large integrated circuits. Moreover, for the development of large models, the need for a graphical representation becomes apparent. It is true that continuous system simulation languages (CSSLs), e.g., ACSL®[4] (*Advanced Continuous System Simulation Language*), offer macro features to build modularly structured models. However, instead of

[4] ACSL, acslX, and PowerBlock are registered trademarks of The AEgis Technologies Group, Inc., 631 Discovery Drive, Huntsville, AL 35806, USA, http://www.acslx.com

writing many thousands of lines of code in a more or less well structured model description language, it is more reliable to hierarchically develop a graphical model using model libraries and to have the result transformed automatically into simulation code. To that end, an easy to use graphical user interface (GUI) alone is not sufficient. What is needed is an appropriate model description language underlying the graphical model representation and its automatic transformation into a modelling language. However, as languages based on the CSSL standard established in 1967 [39] are more simulation languages than modelling languages, they show shortcomings in supporting the development of hierarchical modular structured models. Such a modelling language was described as early as 1979 in Elmqvist's dissertation [15].

On the other hand, in computer science, software projects of ever increasing size have led to the paradigm of Object-Oriented Programming (OOP). In their widely recognised 1991 book, *Object-Oriented Modelling and Design*, J. Rumbaugh and his co-authors present object-oriented modelling as a methodology for the design of large complex software systems that include the analysis of the problem, the design and the software implementation. They use the term *Object Modelling Technique* (OMT) and point out in the preface of their book that object orientation means more than merely a kind of programming. The attractiveness and the success of this fundamental concept has also had an impact on the terminology in physical systems modelling and the manner models of large systems are developed and described. Since about 1990, with the event of modelling languages such as Omola [1], the term *object-oriented physical systems modelling* has come into use. In contrast to classical control, describing a system's behaviour by functional input-output blocks and consequently focusing on the *computational structure*, object-oriented modelling emphasises the view of a model composed of *non-causal* submodels connected accordingly to the *physical structure* of the system. The new approach at that time was promoted by Anderson and Mattson [28] and F. Cellier [10]. In his dissertation [33], Otter used the modelling language Dymola®[5] [15], developed by Emqvist in as early as 1978, for an object-oriented approach to mechatronic systems modelling. Without discussing details of the language Dymola or its successor Modelica®[6] [29], the characteristics of an object-oriented physical systems modelling approach are presented in the following.

Characteristics of Object-Oriented Physical Systems Modelling

According to principles of object-oriented programming in software engineering, object-oriented physical systems modelling may be characterised by the following features.

- Objects
  In object-oriented physical systems modelling, models of components of engineering systems as well as models of physical processes are considered to be

[5] Dymola® is a registered trademark of Dynasim AB, Ideon Science Park, SE-223 70 Lund, Sweden, http://www.dymola.com

[6] Modelica® is a registered trademark of the Modelica Association, http://www.modelica.org

objects. Since basic models are comprised of inherent parameters, e.g., length, mass, moment of inertia, resistance, or data provided from the outside world through so-called *interfaces*, and constitutive equations, there is a correspondence to objects in the sense of object-oriented programming (OOP). Physical system models may be viewed as an aggregation of data and *methods* operating on them. In OOP, the term *method* means a function or a procedure that can process data of a defined type. For instance, the coordinates of a point, a length, and a function that returns the area of a circle around that point may be aggregated into a *class* called *circle*. A circle around a given point of given radius may be named $c1$. Then, $c1$ is a particular object of the class *circle*. Likewise, the voltage across and the current through a resistor, parameters, and the constitutive (non-linear) relation between voltage and current may be considered a class *resistor* corresponding to the element type resistor. A copy of the class *resistor* with given values for the parameters in the constitutive relation called, for instance, $R5$, corresponds to a particular resistor in a circuit.

- Model Hierarchy
  As it is well known, a model of a system may be composed of lower level submodels which in turn may contain submodels as well. That is, physical-system models are *hierarchical* in nature (in the sense of a membership relation).
- Model Classes and Instantiation
  As explained above, physical system models and submodels of components can be viewed as particular objects of a certain class. In object-oriented modelling, the term *instantiation* is adopted from OOP. A model or submodel is called an *instance* of a model class. Models or submodels are *instantiated* from generic models or *model classes* in which more general properties common to the members of a model class are described. The members of a model class have got the same structure and exhibit the same general dynamic behaviour. For example, an instance of the model class *diode* is obtained by giving particular values to its parameters. The resulting instance corresponds to a particular diode in a circuit.
- Inheritance
  If a submodel class is instantiated into a particular submodel, its properties are inherited by the submodel. That is, the particular submodel declaration is a special version of the more general submodel declaration. In Section 11.5.2, an incomplete model class is introduced that only captures the energetic property of a 1-port energy storage element in the sense that this type of element stores a physical quantity such as electrical charge. In that class, a relation between the state and the rate variable is not given. By adding this information, a particular class can be derived. The more special model class inherits all properties of the superclass from which it is derived. Consequently, it describes a particular type of an energy store. Again, by specifying parameters, a particular object or instance of this class is obtained associated with a particular energy store in the system. Another example is an incomplete model class *passiveOnePort* that accounts for the passivity of a 1-port element. A special class *diode* may be obtained from this incomplete superclass by specifying the constitutive relation as that of a diode. Obviously, the subclass inherits the passivity property of a 1-port from the su-

perclass. Using inheritance in the declaration of library model classes has the advantage that the potential of errors in the code is reduced. All features of the superclass are copied. Only features that characterise the more special class need to be added.

- Encapsulation
  Knowledge contained in a model is encapsulated. Only a well defined part of it can be accessed in a well defined manner via interfaces to the outside world of an object. This means that the internal definition of a submodel is not affected by the connection of the submodel to other submodels. The part of a submodel describing its interfaces is separated from the part in which the behaviour is described by means of *non-causal* mathematical equations. The latter do not need to be known when submodels are connected in order to build a hierarchical model. In generalised networks for instance, an interface of a submodel is a pin. It is described by two power variables called *across* and *through* variables.
- Polymorphism
  In a submodel definition, the description of its interfaces to the outside world is separated from the internal description of its dynamic behaviour. In this internal description, multiple possible cases may be taken into account. Consequently, depending on current conditions, the same submodel may exhibit different behaviour.
- Connection of Submodels
  Submodel interfaces are connected accordingly to component interconnections in the real physical system, also called the *physical structure* of the system. In contrary to block diagrams, this means that equations of a submodel must be *non-causal*. The interconnection of submodels may require solving them for certain variables. In block diagrams, it is fixed a priori whether an interface variable of a block is an input or an output variable. This must be taken into account when connecting blocks.

Of course, it has been a tradition in physical systems modelling to build hierarchical models and to connect submodels according to the physical structure long before object-oriented modelling was introduced. In Chapter 11, aspects of object-oriented physical systems modelling will be picked up again when looking at bond graph modelling from an object-oriented modelling point of view.

1.4 Traditional Graphical Model Representations

There are many forms of graphical model representation in use in different engineering disciplines, e.g., free body schematics in mechanical engineering, circuit representations in electrical engineering and in hydraulics, linear graphs as well as block diagrams and signal flow diagrams in control theory. All of these forms are well known and have a long tradition. Therefore, only some aspects of block diagrams and signal flow graphs on the one hand, and network representations on the other hand, will be outlined. The aim is to show the context in which bond graph methodology is embedded. In the following, aspects are discussed from a methodology point of view. In Chapter 11, links to corresponding software tools are considered.

1.4.1 Block Diagrams

Block diagrams have the following characteristics:

- They support the abstraction of unilateral signal flow through a system. Information flow is considered not to be bound to the transfer of energy or the transport of mass. A block in an oriented chain of blocks does not have a direct impact on its predecessor. Feedback is taken into account separately by signal feedback loops.
- The interface variables of a signal block are a priori discriminated into input and output signals, independent of the actual use of a signal block in a block diagram model. This must be taken into account when signal blocks are connected.
- Signal blocks represent functional relations between input and output signals. It is neither required nor ensured that relations comply with first principles of physics. Blocks may represent any linear, or nonlinear algebraic, or time dependent relation.
- Block diagrams display which variables must be known in order to compute others. They represent the structure of the mathematical model, or as van Dixhoorn [45] has termed it, the *computational structure*. They do not reflect the *physical structure* of a system. The reason is that feedback is represented by *separate* feedback loops. Signal blocks cannot be connected like corresponding system components. For instance, if two electrical devices are connected, then the voltages are set to be equal and at the same time, currents are added to zero. In block diagrams, however, a connection between two blocks represents only one signal. As will be pointed out later, in bond graphs, each edge is associated with two conjugate power variables. Consequently, connected bond elements always have a feedback to each other. If submodels in a block diagram are modified by neglecting effects or by taking into account additional ones, then small changes may have a considerable impact on the computational structure and thus on the structure of the block diagram. This disadvantage does not appear in networks or in bond graph graphs.
- As block diagrams represent signal flows and functional relations independent of the physical meaning of variables, they can be used in different engineering disciplines. They are used particularly in control since control systems are often designed in such a manner that there is no feedback between components. The computational structure then corresponds to the physical structure.
- Finally, block diagrams support a hierarchical decomposition into functional blocks.

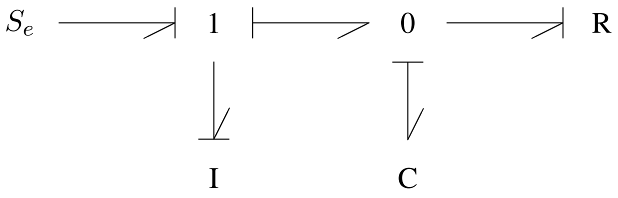

Fig. 1.1 Causally completed bond graph model of a second order system

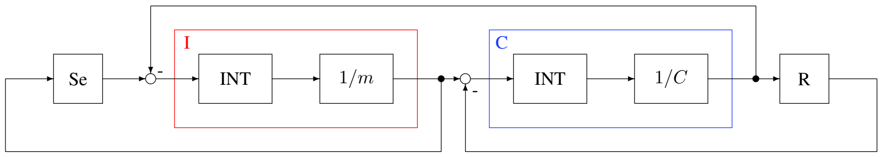

Fig. 1.2 Block diagram corresponding to the bond graph in Figure 1.1

As will become clarified, bond graphs reflect the physical structure of a system, as do networks. On the other hand, the computational structure may be superimposed on a bond graph by adding a perpendicular stroke to each edge, turning the initially non-causal model into a causal one. Moreover, such a causally completed bond graph can be systematically transformed into a block diagram if needed. During this transformation, information about the physical structure gets lost, as can be seen from Figures 1.1 and 1.2. As signal blocks in block diagrams can represent any functional relation, the converse does not hold. Not every block diagram can be transformed into a bond graph. Equations represented by bond graphs should comply with the first principles of physics.

1.4.2 Signal Flow Graphs

In signal flow graphs, the role of edges and vertices is essentially interchanged in comparison to block diagrams. Oriented edges represent functional relations between variables, while nodes are used to represent variables and the summation of variables. In that respect, they may be considered the dual of block diagrams. However, signal flow graphs are less general than block diagrams because besides the summation of variables, no functions with more than one input variable can be represented. Like block diagrams, signal flow graphs represent the computational, not the physical structure of a system. In the case of linear models, they can be used to derive the transfer function between two variables by applying Mason's loop rule [27]. Bond graphs can also be transformed into signal flow graphs as shown in Fig-

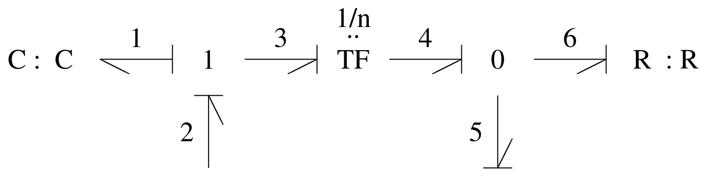

Fig. 1.3 Causally completed bond graph fragment

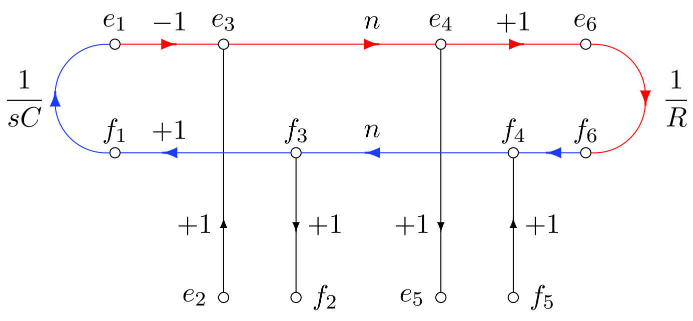

Fig. 1.4 Signal flow graph corresponding to the bond graph fragment in Figure 1.3

ures 1.3 and 1.4. However, as Brown has shown, Mason's loop rule can be applied directly to a bond graph [9].

1.4.3 Networks

In electrical engineering, it is common to represent models as networks. However, this representation is not restricted to electrical systems. By relating the electrical power variables voltage and current to non-electrical quantities appropriately, networks may be used to represent models of systems in other energy domains. A unified approach to the modelling of engineering systems based on so-called *generalised networks* has been introduced by MacFarlane [26]. Networks have the following essential features.

- Contrary to block diagrams, networks represent the physical structure, but not the computational structure of an engineering system. Submodels are connected like corresponding components or devices in the real system. There is no need to decide whether an interface variable is an input or an output variable. The graphical representation of submodels is not uniform across energy domains as in bond graphs, but depends on the engineering discipline. For instance, hydraulic circuits use different icons for submodels than electrical circuits. In any case,

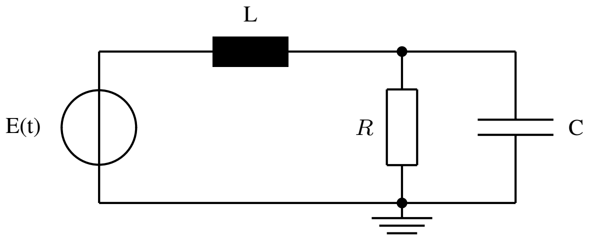

Fig. 1.5 Circuit with two energy stores

Kirchhoff's laws hold for electrical power variables as well as for corresponding variables in other energy domains.
- Networks are hierarchical in nature. Submodels can have a network structure.
- Networks account for energy flows in a system. Circuit nodes connecting pins of submodels comply with power conservation. Moreover, physical quantities, e.g., charge, are conserved if properly taken into account in the development of submodels. Meyer's NMOS transistor model, for instance, does not ensure conservation of charge ([21], Section 3.4.4)

Bond graphs, which will be introduced formally in the next chapter, on the one hand, reflect the physical system structure like networks. On the other hand, a computational structure can be superimposed so that the causally completed bond graph can be considered a concise representation of a block diagram. As bond graphs have features in common with block diagrams and with networks, both representations have been briefly discussed in this introduction. Moreover, bond graphs can be considered a core model representation for the following reasons.

- Generalised networks can be systematically converted into bond graphs (cf. Section 2.7). If orientations of edges in a bond graph indicating the reference direction of the energy flow across a bond are chosen with care, then the directed bond graph is equivalent to the network from which it has been constructed (Section 2.8). For instance, if the circuit in Figure 1.5 is converted into a bond graph, then the result is the bond graph in Figure 1.3.
- From a causally completed bond graph, a block diagram as well as a signal flow graph can be derived.
- Finally, domain-specific iconic diagrams can be systematically converted into a bond graph if there are bond graph equivalents of basic icons.

1.5 Conclusion

Dynamic system modelling has a long tradition. On the other hand, the views of model developers, methods and corresponding software tools change, and new issues and new applications have been tackled. Due to the ever increasing power of computers and even of mobile computers, imitation, or the prediction of reality is becoming steadily less expensive in more and more fields. Even if experiments with the real system under consideration are feasible, less expensive computer simulation will reduce them to some extent. In this context, many publications on dynamic systems modelling have already appeared and many more are to be expected in the future. This introduction has focused on some general aspects of dynamic systems modelling. Essential features of block diagrams and of networks have been briefly recalled because bond graphs combine characteristics of both graphical model representations. In summary, one can say:

- Graphical model representation is a means of communication between humans and between humans and computer programs. The choice of an appropriate model representation depends on the purpose of the modelling process, the questions it helps to answer.
- In the process of abstraction, some properties of a real system and some effects are taken into account in an idealised manner, while others are completely neglected. The assessment of properties and their selection is guided by the purpose of the modelling task and by the experience of the model developer. Consequently, graphical model representations always reflect only some aspects. Thus, block diagrams appropriate for representing signal processing represent the computational structure, but not the physical structure of a system, whereas on the other hand, electrical, or hydraulic networks display how corresponding real system components are connected at the expense of the computational structure.

Graphical model representations are particularly appropriate for some purposes while they are less suited for others. Some may be converted into others, e.g., a network of an electric circuit can be converted into a block diagram. A possible reason for such a conversion may be that the available modelling software does not support a combination of different model representations. The choice of a graphical model representation formalism in dynamic modelling is similar to the choice of a programming language in a software development project. Depending on the task, some programming languages are more suited than others. Often, several programming languages may serve the requirements. Moreover, some programming languages can be translated automatically into others. Since graphical model representation formalisms are not equally suited for all applications and purposes, it may be reasonable to use a combination of them.

Bond graphs fit into the spectrum of graphical model representations as a formalism that is particularly suited for engineering systems with effects from multiple energy domains. Consequently, they are an ideal representation for mechatronic and for micro-mechanical systems. Moreover, they not only support model development as some kind of visual language, they have also been proven as an appropriate core representation. That is, several other graphical representations can be converted into bond graphs, while from causal bond graphs graphical representations, e.g., block diagrams, equations, transfer functions and other information can be derived systematically. This motivates the introduction of bond graph based physical systems modelling in the next chapter.

References

[1] M. Andersson. Omola – An Object Oriented Language for Model Representation. Master's thesis, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1990. Licentiate thesis TFRT-3208.

[2] P. Borne, G. Dauphin-Tanguy, J.P. Richard, F. Rotella, and I. Zambettakis. *Modélisation et Identification des Processus*, volume tome 2. Éditions Technip, Paris, 1992.

[3] W. Borutzky. *Bond graphs – A Methodology for Modelling Multidisciplinary Dynamic Systems*. Frontiers in Simulation. SCS European Publishing House, Erlangen, Ghent, 2000. ISBN: 1-56555-183-4 (in German).

[4] W. Borutzky, editor. *Special Issue Bond Graph Modelling*, volume 17/1 of *Simulation Modelling Practice and Theory*, 2009. Elsevier.

[5] W. Borutzky, A. Orsoni, and R. Zobel, editors. *Proc. of the 20th European Conference on Modelling and Simulation*, 2006. European Council for Modelling and Simulation. Hard Copy: ISBN: 0-9553018-0-7, CD-ROM: ISBN: 0-9553018-1-15.

[6] A.M. Bos and P.C. Breedveld. Update of the Bond Graph Bibliography. *Journal of the Franklin Institute*, 319(1/2):269–286, Jan./Feb. 1985.

[7] P.C. Breedveld and J. van Amerongen. *Dynamische systemen: modelvorming en simulatie met bondgrafen*. Open universiteit, Heerlen, The Netherlands, 1994.

[8] P.C. Breedveld, R.C. Rosenberg, and T. Zhou. Bibliography of Bond Graph Theory and Application. *Journal of the Franklin Institute*, 328(5/6):1067–1109, 1991.

[9] F.T. Brown. Direct Application of the Loop Rule to Bond Graphs. *Journal of Dynamic Systems, Measurement and Control*, pages 253–261, September 1992.

[10] F.E. Cellier. *Continuous System Modeling*. Springer-Verlag, New York, Berlin, Heidelberg, 1991. ISBN: 0-387-97502-0.

[11] F.E. Cellier. Bond Graphs: The Right Choice for Educating Students in Modeling Continuous-Time Physical Systems. *SIMULATION*, 64(3):154–159, March 1995.

[12] F.E. Cellier. World Wide Web – The Global Library: A Compendium of Knowledge About Bond Graph Research. In J.J. Granda and G. Dauphin-Tanguy, editors, *1997 International Conference on Bond Graph Modeling and Simulation (ICBGM '97)*, volume 29(1) of *Simulation Series*, pages 187–191. SCS, 1997. URL http://www.inf.ethz/personal/fcellier/BondGraphs/bg.html.

[13] G. Dauphin-Tanguy. *Les bond graphs*. Hermes Science Europe Ltd., Paris, France, 2000. ISBN: 2-7462-0158-5.

[14] P. Dransfield. *Hydraulic Control Systems – Design and Analysis of Their Dynamics*. Springer-Verlag, New York, 1981.

[15] H. Elmqvist. *A Structured Model Language for Large Continuous Systems*. PhD thesis, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1978. Report CODEN: LUTFD2/(TFRT-1015)/1-226/(1978).

[16] P.J. Gawthrop and G.P. Bevan. Bond Graph Modeling. *IEEE Control Systems Magazine*, pages 24–45, April 2007.

[17] P.J. Gawthrop and S. Scavarda, editors. *Special Issue on Bond Graphs*, volume 216(1) of *Proceedings of the Institution of Mechanical Engineers Part I: Journal of Systems and Control Engineering*, London, UK, 2002. Professional Engineering Publishing.

[18] P.J. Gawthrop and L. Smith. *Metamodelling: Bond Graphs and Dynamic Systems*. Prentice Hall International (UK) Limited, Hemel Hempstead, 1996. ISBN: 0-13-489824-9.

[19] V. D. Gebben. Bond Graph Bibliography. *Trans. ASME Journal of Dynamic Systems, Measurement, and Control*, 99(2):143–145, 1977.

[20] J.J. Granda. *Computer-aided modelling program (CAMP): a bond graph preprocessor for computer-aided design and simulation of physical systems using digital simulation languages*. PhD thesis, Univ. of California, Davis, 1982.

[21] E.-H. Horneber. *Simulation elektrischer Schaltungen auf dem Rechner*. Springer-Verlag, 1985.

[22] D.C. Karnopp and R.C. Rosenberg. *Analysis and Simulation of Multiport Systems – The Bond Graph Approach to Physical System Dynamics*. MIT Press, Cambridge, MA, 1968.

[23] D.C. Karnopp and R.C. Rosenberg. *System Dynamics: A Unified Approach*. John Wiley & Sons, Inc., New York, 1975.

[24] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. *System Dynamics - Modeling and Simulation of Mechatronic Systems*. John Wiley & Sons Inc., Third edition, 2000. ISBN 0-471-33301-8.

[25] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. *System Dynamics - Modeling and Simulation of Mechatronic Systems*. John Wiley & Sons Inc., Fourth edition, 2005. ISBN: 0-471-70965-4.

[26] A.G.J. MacFarlane. *Engineering Systems Analysis*. Harrap, London, UK, 1964.

[27] S. J. Mason. Feedback Theory: some properties of signal flow graphs. In *Proc. IRE 41*, pages 1144–1156, 1953.

[28] S.E. Mattsson. Object-Oriented Modelling of a Real Continuous-Time System. In J. Stephenson, editor, *Proc. of the 1992 European Simulation Multiconference*, pages 241–245, San Diego, CA, 1992. SCS. York, UK, June 1992.

[29] Modelica Association. Modelica and the Modelica Association. URL http://www.modelica.org.

[30] A. Mukherjee and R. Karmakar. *Modelling and Simulation of Engineering Systems through Bondgraphs*. Narosa Publishing House, New Delhi, India, 2000. ISBN: 81-7319-279-0.

[31] A. Mukherjee, R. Karmakar, and A.K. Samantaray. *Bond Graph in Modeling, Simulation and Fault Identification*. I.K. International Publishing House, New Delhi, India, 2006. ISBN: 81-88237-96-5.

[32] S.T. Nannenberg. *Bondgraaftechniek*. Delta Press, Overberg, Niederlande, 1987.

[33] M. Otter. *Objektorientierte Modellierung mechatronischer Systeme am Beispiel geregelter Roboter*. PhD thesis, Ruhr-Universität Bochum, 1994.

[34] H.M. Paynter. *Analysis and Design of Engineering Systems*. M.I.T. Press, Cambridge, Massachusetts, USA, 1961.

[35] H.M. Paynter. An Epistemic Prehistory of Bond Graphs. In P.C. Breedveld and G. Dauphin-Tanguy, editors, *Bond Graphs for Engineers*, pages 3–17. North-Holland, 1992.

[36] G. Romero, J. Félez, J. Maroto, and J.M. Mera. Efficient simulation of mechanism kinematics using bond graphs. *Simulation Modelling Theory and Practice*, 17/1:293–308, 2009.

[37] A.K. Samantaray and B. Ould Bouamama. *Model-based Process Supervision – A Bond Graph Approach*. Advances in Industrial Control. Springer, London, 2008. ISBN 978-1-84800-158-9.

[38] A. Schöne. Systembeschreibung durch simultane Diagramme. *Regelungstechnik*, 24(2):37–72, 1976.

[39] J.C. Strauss, D.C. Augustin, M.S. Fineberg, B.B. Johnson, R.N. Linebarger, and F.H. Sanson. The SCi Continuous System Simulation Language (CSSL). *SIMULATION*, pages 281–303, Dec. 1967.

[40] J.U. Thoma. *Simulation by Bondgraphs – Introduction to a Graphical Method*. Springer-Verlag, New York, 1990.

[41] J.U. Thoma and B. Ould Bouamama. *Modeling and Simulation in Thermal and Chemical Engineering (A Bond Graph Approach)*. Springer-Verlag, Berlin, 2000.

[42] J.U. Thoma and H.J. Halin, editors. *Special Issue Bondgraphs for Modeling and Simulation*, volume 7(5–6) of *Simulation Practice and Theory*, 1999. Elsevier.

[43] J.U. Thoma and G. Mocellin. *Simulation with Entropy in Engineering Thermodynamics*. Springer, Berlin, Heidelberg, New York, 2006. ISBN -10 3-540-32798-3.

[44] I. Troch, W. Borutzky, and P.J. Gawthrop, editors. *Mathematical & Computer Modelling of Dynamical Systems, Special Issue: Bond Graph Modelling*, volume 12 (2–3). Taylor & Francis, April-June 2006.

[45] J.J. van Dixhoorn. Bond graphs and the challenge of a unified modelling theory of physical systems. In F.E. Cellier, editor, *Progress in Modelling and Simulation*, pages 207–245. Academic Press, New York, 1982.

[46] M. Vergé and D. Jaume. *Modélisation structurée des systèmes avec les Bond Graphs*. Edition Technip, 2003. ISBN: 2-7108-0838-2.