Constraint Logic Programming for Structure-Based Reasoning about Dynamic Physical System*

Yousri El Fattah
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Wien, Austria
Phone: +43-1-5336112
Fax: +43-1-5320652
Email: yousri@ai.univie.ac.at

May 17, 1994

Abstract

The paper describes a framework for reasoning about dynamic physical systems based on structure. The framework integrates the language of bond graphs (BG) with the language of constraint logic programming (CLP). The advantage of such integration is twofold. First, to exploit the wealth of reasoning methods developed in the BG area within system dynamics. Second, to enhance the naturalness of representation of system relations and possibly increase solution efficiency via CLP. The paper describes methods for causal modeling of dynamic physical system including the generation of causal explanations.

*This work was supported by the Austrian "Fonds zur Förderung der wissenschaftlichen Forschung" under grant P9426-PHY. Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Science and Research.

1 Introduction

The bond graph is a powerful, versatile method for modeling dynamic physical systems of multidisciplinary nature. The method was created by Henry Paynter, who believed that energy and power alone are the fundamental dynamical variables, and that such variables are adequate to represent all connections and interactions between such systems [4]. Since then, work on bond graphs has been rapidly expanding, and a part of that work has been devoted to the study of causality[8]. Causality plays important roles in modeling, analysis, and design of dynamic physical systems [9]. Bond graphs also offer a knowledge representation framework useful for qualitative reasoning, and in particular for representing and generating causal order in a physical system [13, 15, 14]. Causality has been a subject of debate in qualitative physics [5, 1, 2, 6]. Part of that debate stems from the type of representation used for modeling a physical system (differential equations in [5] and confluences in [1]) and disagreement over which variables are exogenous. In bond graphs, causality assignment rules are grounded in basic physical insight [4], and representation is in terms of prototypical physical elements (mechanisms). Bond graph causality is viewed as "an aid in computing bond graph model behavior and as a guide to modelers seeking insight directly (i.e., without having to obtain the system equations explicitly)" [9]. Causality of a bond graph aims at satisfying the basic element causal properties at the system, or bond graph level. It can be computed by the so-called Sequential Causality Assignment Procedure (SCAP) [7].

This work represents bond graphs in the language of constraints and shows how constraint-based programming can be used for bond graph modeling. The causality conditions for bond graph elements and junctions are stated as constraints, and causal labeling of a bond graph is viewed as a constraint satisfaction problem. Using that constraint formulation SCAP is re-implemented using the language of constraint logic programming in the boolean domain, CLP($\mathcal{B}$). The advantage of such implementation is to enhance the naturalness of representation of causal relations, and to possibly increase solution efficiency. The paper also shows how block diagrams can be computed from a causally-ordered bond graph. A block diagram depicts the interaction between the various mechanisms that produce the system output from the inputs and identifies the feedback loops in the system model.

The paper is organized as follows. Section 2 describes a system architecture and motivates this work. Section 3 represents bond graphs in the language of constraints. Section 4 formulates causal labeling of a bond graph as a constraint satisfaction problem and gives a CLP($\mathcal{B}$) procedure. Section 5 describes a method for causal modeling, and section 6 gives general conclusions.

2 Structure-based Reasoning

Fig. 1 presents a computing architecture for structure-based reasoning about dynamic physical systems. The input is a structural description of a dynamic physical system and queries about its behavior. The output are answers to the queries including explanations. The computing tasks are the following,

1. translating structural representation to bond graph representation,

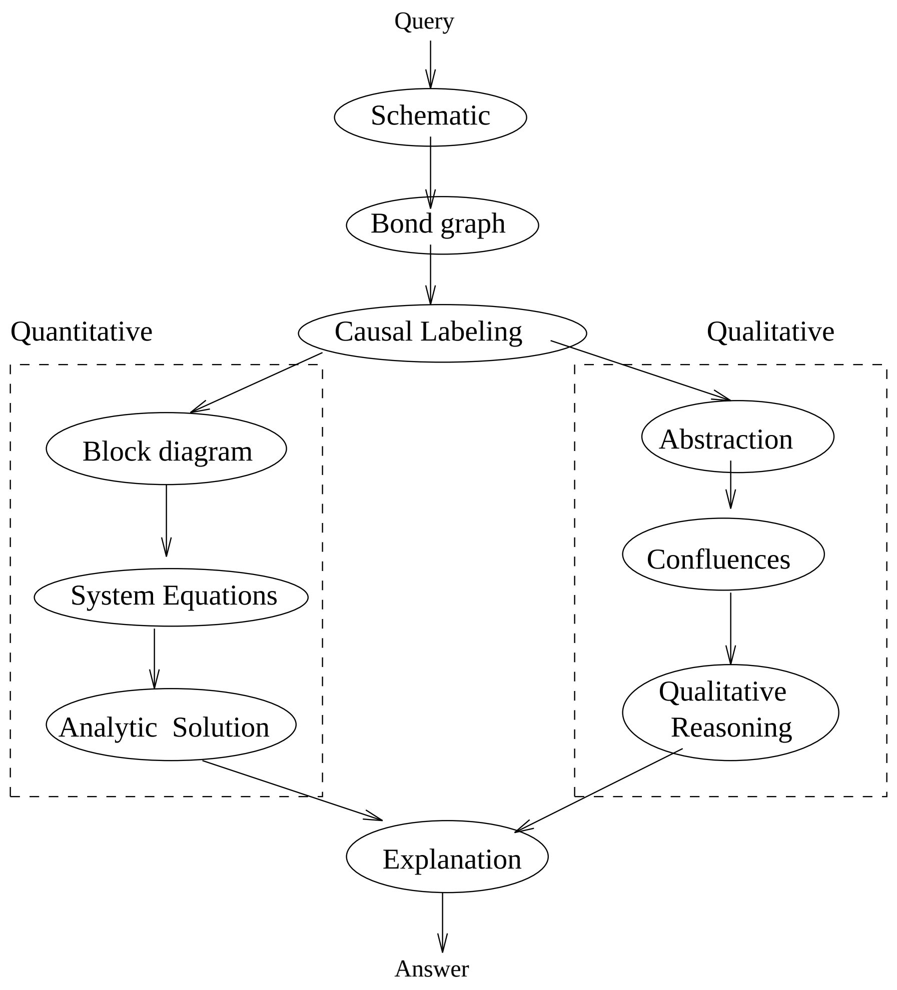


Figure 1: System architecture for structure-based reasoning.

2. causal labeling of the bond graph,
3. quantitative/qualitative reasoning,
4. generating answers/explanations to queries.

Queries about dynamic system's behavior depend on the type of problem solving involved, such as, prediction, analysis, diagnosis, and design. Examples of queries for the simple mass spring system shown in fig. 2 are as follows:

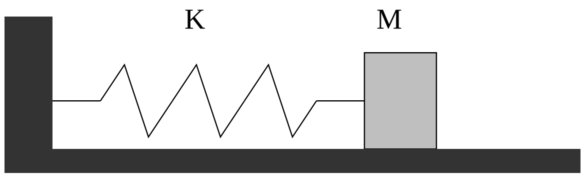


Figure 2: A simple mass spring system.

Prediction. What happens if the mass is extended and then released?

Analysis. Is the system stable?

Diagnosis. Why is the system oscillating rapidly when exited at some frequency?

Design. What parameters will ensure specific oscillation?

Reasoning in the proposed framework is a hybrid of quantitative and qualitative. See fig. 1. Quantitative reasoning is based on analytical or numerical solutions of the system equations. System equations may be in the form of differential equations or in the form of a transfer function in the time or frequency domain. Those equations are based on the system's block-diagram. Qualitative reasoning is useful when system order reduction is desired or knowledge of exact values is unavailable. Qualitative reasoning involves abstractions of system equations in the quantity space (Q-space). Confluences are qualitative differential equations [1] defined in the ($\pm$) Q-space. Qualitative reasoning may be based on time-abstraction by identifying slow and fast variables, which has been used for quantitative model reduction [11].

2.1 Why Bond Graphs?

Translating structural representation, usually in the form of system schematic, to a bond graph allows canonical representation of dynamic systems from disparate domains. The bond graph language is increasingly accepted and used in various engineering disciplines, and offers a flexible efficient framework for automated modeling. According to [10]:

> You can construct a bond graph model to reveal any physical level of detail desired; conversely, you can mask as much of the details as you deem unnecessary by suitable structuring of the model. Exploiting this feature leads to models that are posed to answer questions about system behavior very efficiently.

2.2 Why causal modeling?

Questions may be raised as of why bother about causal modeling. Why not directly represent the system components by their mathematical set of equations (constraints), and then solve those equations to determine system's behavior?

Causal modeling provides an additional source of knowledge that greatly enhances the modeling process and increases the efficiency of the reasoning tasks. See [3, 5, 12]. Rosenberg [9] discusses how causality can be exploited to:

- Check correctness of a proposed design,
- Guide the engineer in assembling sub-models into a system model,
- Suggest suitable test conditions for fault diagnosis, and
- Analyze interaction between nonlinearities and uniqueness of system response.

In our framework to structure-based reasoning causal labeling is crucial to generate block diagrams, causal explanations, identify state variables and system order, and derive system equations. Although causal modeling may not be so important for prediction, it is a key for qualitative reasoning, for analysis, and diagnostic reasoning.

3 Bond Graphs

This section shows how bond graphs can be represented in the language of constraints. A bond graph (BG) consists of a set of nodes (called *multiport subsystems*) and a set of edges (called *bonds*). A 1-port subsystem can be one of the following types: a *flow source* (SF), an *effort source* (SE), a *capacitor* (C), an *inertia* (I), or a *resistor* (R). The type of a 2-port subsystem can be a *transformer* (TF), or a *gyrator* (GY). Other types of *N-port* subsystems ($N \geq 2$) are *junctions*: a *common-effort junction* (0), and a *common-flow junction* (1). An N-port is by definition connected to other multiports by a set of N bonds, $\mathcal{B}$. $\mathcal{B}$ may be partitioned to $\mathcal{B}^+$ and $\mathcal{B}^-$, defined as the subsets of incoming (directed toward the port) and outgoing (directed away from the port) bonds, respectively. A pair of multiports linked by a bond are constrained to share the same variables. Each bond B has 4 variables: effort e^B, flow f^B, displacement q^B and momentum p^B. Two of the variables, namely the displacement and the momentum (called the energy variables) are the time derivatives of the flow and effort variables, respectively. The effort and flow variables are called the power variables. Table 1 states the constraints on the bond variables for each type of multiport. Constraints on the one port elements (R, C, I) are assumed linear in the table, although any appropriate relation between the referenced bond variables may be considered instead.

Table 1: Constraints imposed by various multiport types on the energy/power variables of their linking bonds.

Multiport		*Bonds* $\mathcal{B}$	*Constraints*
Type	*Parameters*		
SE	e	B	$e^B = e$
SF	f	B	$f^B = f$
R	r	B	$e^B = r \cdot f^B$
C	c	B	$q^B = c \cdot e^B$
I	i	B	$p^B = i \cdot f^B$
TF	m	B_1,B_2	$e^{B_1} = m \cdot e^{B_2}$, $f^{B_2} = m \cdot f^{B_1}$
GY	k	B_1,B_2	$e^{B_1} = k \cdot f^{B_2}$, $e^{B_2} = k \cdot f^{B_1}$
1		$B_1, \ldots, B_N$	$\sum_{B \in \mathcal{B}^+} e^B = \sum_{B' \in \mathcal{B}^-} e^{B'}$, $f^{B_1} = \ldots = f^{B_N}$
0		$B_1, \ldots, B_N$	$\sum_{B \in \mathcal{B}^+} f^B = \sum_{B' \in \mathcal{B}^-} f^{B'}$, $e^{B_1} = \ldots = e^{B_N}$

Methods described here for causal modeling do not require linear relations and produce results for general nonlinear relations. See section 5. However, if nonlinear relations are considered, interaction between nonlinearities may impose constraints on causal directions to ensure uniqueness of system response. This may introduce modifications to table 2 and possibly procedure 1 for causal ordering.

4 Causality

A binary variable, C_B^M, called *causality variable* is defined for each multiport relative to its connecting bonds in the graph. A multiport M is said to have an *effort causality* along a

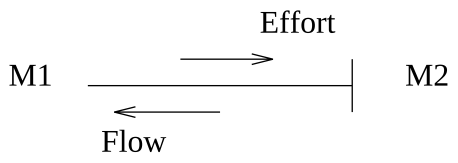


Figure 3: Graphical representation of effort and flow causalities on a bond B depicted using bond graphs notation; $M1$ has effort causality $C_B^{M_1} = 1$, while $M2$ flow causality $C_B^{M_2} = 0$.

bond B (expressed as $C_B^M = 1$) if M takes the flow in B as input and produces the effort in B as output. A multiport M is said to have a *flow causality* along a bond B (expressed as $C_B^M = 0$) if M takes the effort in B as input and produces the flow in B as output. The causality assignment on a bond is represented graphically in fig. 3 using the bond graph notation [7]. According to that notation causality is shown by adding a short bar to the end of a bond with the understanding that effort causality exists in the direction towards the bar and flow causality away from it.

Causality of a pair of multiports M_1, M_2 linked by a bond B must be complementary, i.e., $C_B^{M_1} \equiv \neg C_B^{M_2}$. This corresponds to arc constraint on causality variables (table 2). Node constraints on causality variables are those to be satisfied by each multiport M according to type as in table 2. See [7]. A constraint for a multiport is defined by the set of value tuples that are allowed for its causality variables. Source elements have one causality variable with only one possible value; 1 for type SE and 0 for type SF. Elements of type R impose no constraint; both binary values are allowed. Elements of type C, I, called *storage* elements, have two sorts of constraints: *integral* and *derivative.* Integral (derivative) causality means that the directional (input-output) relation between the power variables– effort and flow– for the storage element has an integral (derivative) form. For a capacitor C, integral (derivative) causality exists when the flow (effort) is the input and the effort (flow) is the output. For an inertia I, integral (derivative) causality exists when the effort (flow) is the input and the flow (effort) is the output. The 2-port element TF must assign different values to the causality variables of its two bonds,i.e., only tuples 01 and 10 are allowed. The 2-port element GY must assign same value to the causality

Table 2: Constraints on causality of bonds

Node constraint (by type of mutiport M)		
Type	*Bonds $\mathcal{B}$*	*constraint*
SE	B	C_B^M
SF	B	$\neg C_B^M$
R	B	$C_B^M \vee \neg C_B^M$
C	B	C_B^M (integral); $\neg C_B^M$ (derivative)
I	B	$\neg C_B^M$ (integral); C_B^M (derivative)
TF	B_1,B_2	$C_{B_1}^M \oplus C_{B_2}^M$
GY	B_1,B_2	$C_{B_1}^M \equiv C_{B_2}^M$
0	$B_1, \ldots, B_N$	$\neg C_{B_i}^M \equiv \bigwedge_{j \neq i} C_{B_j}^M$
1	$B_1, \ldots, B_N$	$C_{B_i}^M \equiv \bigwedge_{j \neq i} \neg C_{B_j}^M$
Arc constraint		
Bond	*Connected multiports*	*constraint*
B	M_1, M_2	$C_B^{M_1} \oplus C_B^{M_2}$

variables of its two bonds,i.e., only tuples 00 and 11 are allowed. The N-port junction of type 0 (called common-effort junction) can have only one of its causality variables assigned value 0 with the remaining N-1 variables be assigned value 1, i.e., only N tuples are allowed $111\cdots110, 111\cdots101, \ldots, 101\cdots111, 011\cdots111$. The N-port junction of type 1 (called common-flow junction) can have only one of its causality variables assigned value 1 with the remaining N-1 variables be assigned value 0, i.e., only N tuples are allowed $000\cdots001, 000\cdots010, \ldots, 010\cdots000, 100\cdots000$.

The task of causal labeling for a bond graph BG consists in assigning values to causal variables C_B^M consistent with all node and arc constraints with preferred integral causality being assigned to maximum number of storage elements in BG. This is done by the algorithm SCAP [7], which performs sequential constraint satisfaction. SCAP proceeds sequentially by first computing a causal ordering consistent with all the sources. If that step succeeds then the consistent labeling is further extended to include as many integral causality constraints for storage elements as possible. If integral causality is satisfied for all the storage elements then the causality variables may all be assigned values, or some of them remain unassigned. In the latter case, which indicates the presence of algebraic loops, a causality variable may be assigned an arbitrary value. Then consistent labeling is further extended, and the process continues until all the variables are instantiated. Using the constraint formulation of table 2 SCAP was re-implemented in the language of constraint logic programming in the boolean domain, CLP($\mathcal{B}$). See procedure 1. The advantage of such implementation is to enhance the naturalness of representation of causal relations, and possibly increase solution efficiency.

Procedure 1 compute_causal_order

Input: Bond graph BG

Output: Causality assignments and their types.

1. Let $\sum$ denote the set of causal assignment by source elements. Let the set of junction constraints be $\mathcal{T}$.

2. If $\sum \cup \mathcal{T}$ is unsatisfiable then BG has *invalid causality* and return.

3. Else let σ be the causal assignment entailed by $\sum \cup \mathcal{T}$ and do:

    (a) While one-port M exists of type C or type I do:

    If γ_M^{integ} is the integral causality constraint for M and $\sigma \cup \gamma_M^{integ} \cup \mathcal{T}$ is satisfiable by σ' then $\sigma \leftarrow \sigma'$ else BG has *derivative causality.*

(b) If BG has not been assigned derivative causality then BG has *integral causality*.

If more than one assignment σ exists, then BG has *algebraic loops*.

5 Block Diagram

In this section the problem of computing a block diagram from a causally-ordered bond graph is considered. A block diagram depicts the interaction between the various mechanisms that produce the system output from the inputs and identifies the feedback loops in the system model.

A block diagram in our computer program is a node-labeled tree, called BD-tree, whose nodes are variables and labels are function specifications. A function specification includes the definition of a function or operator and possibly a list of parameters. Operators can be differential operators: integration **integ** and differentiation **deriv**, or algebraic: summation **sum**, multiplication **mult**, and division **div**. Function definitions include **e2q** for one-port elements of type C, **f2e** for one-port elements of type R, and **f2p** for one port elements of type I. The function **e2q** for element of type C specifies displacement as a function of effort and is in its linear form given in terms of a capacitance parameter as in table 1. The function **f2e** for element of type R specifies effort as a function of flow and is in its linear form given in terms of a resistance parameter as in table 1. The function **f2p** for element of type I specifies momentum as a function of flow and is in its linear form given in terms of an inductance parameter as in table 1. Implicit definitions of **q2e, e2f, p2f** for types C, R, I are obtained by inverting the respective definitions **e2q, f2e, f2p**.

Two port elements of types TF, GY are specified in terms of constants called transformer modulus **m** and gyrator modulus r, respectively. Other function specifications are: **eq, input, feedback**. The label **input** denotes a dummy function that takes no arguments and signifies that a variable is independent (i.e., exogenous). The functions **eq, feedback** take one argument and signify equality. **eq** is reserved for internal nodes of the tree while **feedback** is reserved for the leaf nodes. The specification **input** is also reserved for the leaf nodes. The specification **sum** requires a list of sign parameters and means the bottom-up (or, left-to-right) signed summation of the children variables.

A BD-tree has the following semantics. Given a node V with the label F and children list CL, the value of V is the output of the function F when applied to the value list of the children CL, written $F\#CL$. For example, in the BD-tree shown in fig. 4(d), the value of variable f^7 (root of the tree) is obtained by applying the function specification $p2f(inertia)$ to p^7. p^7 is the time integral of its child node e^7, which equals $e^5 - e^6$.

Procedure 2 describes how BD-tree can be computed. The procedure first computes the set of directional constraints, DC, determined by the causal order and the bond graph constraints. DC is a list of elements in the form, $V \leftarrow F\#CL$. The procedure computes the tree top-down by associating the label F to the node V and computes recursively the subtrees for each child in CL. The procedure maintains a list $EXCLUDE$ to identify feedback loops. If a node with the **eq** label has a child that is a member of $EXCLUDE$ then a feedback loop exists and in such case the node is assigned a feedback label. If a node has no children then the node correspond to an exogenous (input) variable.

Procedure 2 compute_block_diagram

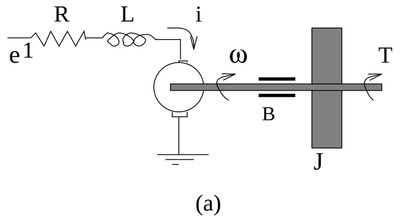
Input: Bond graph BG, Causal order D, bond variable V.

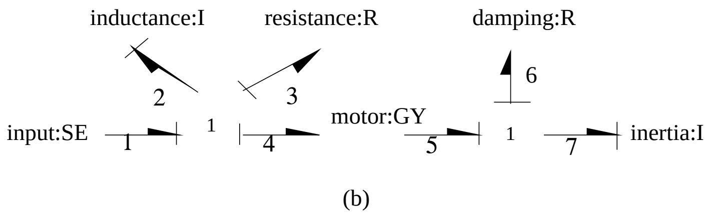
Output: Block diagram in the form of a node-labeled tree.

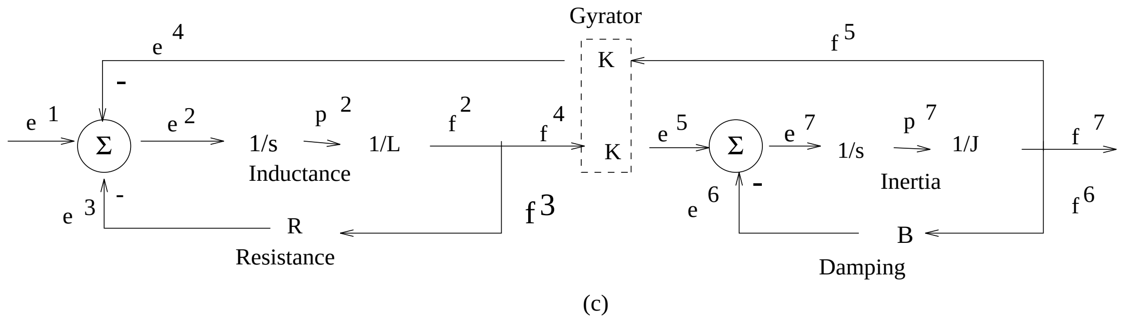
Initialize: List $EXCLUDE$ to empty.

```
1. Compute the list of directional constraints $DC$ for all multiports in $BG$.
2. If $V \leftarrow \sum_{i=1}^{n} S_i \cdot V_i$ is in $DC$ then add $V$ to $EXCLUDE$ and return a tree rooted at $V$ labeled $sum : [S_1, \ldots, S_n]$ $(S_i \in \{+,-\})$ with children subtrees $T(V_i)$ obtained by recursing.
3. If $V \leftarrow F\#V'$ is in $DC$ then do:
   (a) IF $F = eq$ and $V' \in EXCLUDE$ then return the single node tree $V$ labeled $feedback(V')$.
   (b) Else add $V$ to $EXCLUDE$ and return a tree rooted at $V$ labeled $F$ with the subtree $T(V')$ obtained by recursing.
4. Else return the single node tree $V$ labeled $input$.
```

Example 1 Consider the motor system shown in fig. 4 (a). The causal bond graph is shown in fig. 4 (b). The block diagram is shown in fig. 4 (c), where the block $1/s$ represents integration (s is the Laplace transform). The symbolic representation of the block diagram generated by our program is shown in fig. 4 (d), which reads as follows. The rotational speed f^7 depends on the momentum p^7 which is the integral of the torque e^7. The torque e^7







```
f^7:p2f(inertia)
    p^7:integ
        e^7:sum:[+,-]
            e^5:mult:r(motor)
                f^4:eq
                    f^2:p2f(inductance)
                        p^2:integ
                            e^2:sum:[+,-,-]
                                e^1:input
                                e^3:f2e(resistance)
                                    f^3:feedback(f^2)
                                e^4:mult:r(motor)
                                    f^5:feedback(f^7)
            e^6:f2e(damping)
                f^6:feedback(f^7)
```

(d)

Figure 4: Electric motor drive; (a) schematic diagram, (b) augmented bond graph, (c) block diagram, (d) symbolic block diagram for the output f^7 corresponding to the rotational speed of the motor ω. The superscript for a bond variable corresponds to the bond label.

is the difference between the motor torque e^5 and the torque e^6 dissipated by the damping. e^5 is proportional to the armature current f^4 which equals the current f^2 flowing in the inductance. f^2 is proportional to the flux linkage p^2 which is the integral of the voltage e^2. e^2 equals the input voltage e^1 minus the voltage drop across the resistance e^3 minus the back electro motive force e^4 which is proportional to the rotational speed f^7.

6 Concluding Remarks

The bond graph method provides a versatile framework for dynamic system modeling and for reasoning about dynamic system behavior based on structure. In spite of their utility, bond graphs are not widely used in artificial intelligence. This may be attributed partly to the differences in the languages and techniques used in both areas. One motivation for this paper is to embed the bond graph method in the framework of constraint-based representation and reasoning. By doing this, constraint-based programming and computational methods known in artificial intelligence become readily available for bond graphs. Another motivation is that by merging the areas of constraint-based programming and bond graphs, we hope to spur new developments in the area of qualitative physics. The paper presents the results of initial efforts toward those goals. First result of this work is formulating the problem of causal ordering of a bond graph in the language of constraints and implementing the well-known sequential causality assignment procedure (SCAP) in the language of constraint logic programming (CLP). The contribution of this result is twofold: (a) enhanced naturalness of representation of causal relations, and (b) possible increase in solution efficiency. Second result is describing a method for computing block diagrams from bond graphs. A block diagram depicts the interaction between the various mechanisms that produce the system output from the inputs and identifies the feedback loops in the system model. Once a block diagram is produced, the transfer function of a linear system can be derived which allows predictions in the time or frequency domain.

Acknowledgements

I thank Robert Trappl, Igor Mozetič and Christian Holzbaur for discussions and comments on this work. Thanks also to Jan Top for advice and for pointers to relevant papers and books.

References

[1] J. de Kleer and J. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[2] J. de Kleer and J.S. Brown. Theories of causal ordering. *Artificial Intelligence*, 29:33–62, 1986.

[3] R. Dechter and J. Pearl. Directed constraint networks: A relational framework for causal modeling. In *Proceedings, IJCAI-91*, pages 1164–1170, 1991.

[4] H.M.Paynter. *Analysis and Design of Engineering Systems.* MIT Press, Cambridge, MA, 1961.

[5] Y. Iwasaki and H.A. Simon. Causality in device behavior. *Artificial Intelligence*, 29:3–32, 1986.

[6] Y. Iwasaki and H.A. Simon. Theories of causal ordering—reply to de kleer and brown. *Artificial Intelligence*, 29:63–67, 1986.

[7] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. *System Dynamics: A Unified Approach.* John Wiley& Sons, New York, second edition, 1990.

[8] J. Montbrun-Di Flippo, M. Delgado, C.Prie, and H.M.Paynter. A survey of bond graphs: Theory, applications and programs. *Journal of the Franklin Institute*, 328(5/6):565–606, 1991.

[9] R. C. Rosenberg. Exploiting bond graph causality in physical system models. *Transactions of the ASME: Journal of Dynamic Systems, Measurement and Control*, 109:378–389, 1987.

[10] R. C. Rosenberg. Reflections on engineering systems and bond graph. *Transactions of the ASME: Journal of Dynamic Systems, Measurement and Control*, 115:242–251, 1993.

[11] C. Sufur and G. Daphin-Tanguy. Bond graph approach to multi-time scale system analysis. *Journal of the Franklin Institute*, 328(5/6):1005–1026, 1991.

[12] J. Top. In P.C. Breedveld and G. Dauphin-Tanguy, editors, *Bond Graphs for Causal Explanations*, pages 313–321. North-Holland, Amsterdam, 1992.

[13] J. Top and H. Akkermans. Processes as components: on the primitives of qualitative scientific physics. In B. Neumann, editor, *Proceeding of the 8th European Conference on Artificial Intelligence (ECAI 90)*, pages 643–648, New York, 1990. John Wiley & Sons.

[14] J. Top and H. Akkermans. Computational and physical causality. In *Proceedings, IJCAI-91*, pages 1171–1176, 1991.

[15] J. Top and H. Akkermans. Qualitative reasoning about physical systems: an artificial intelligence perspective. *Journal of the Franklin Institute*, 328(5/6):1047–1065, 1991.