BOND GRAPH MODELING OF VARIABLE STRUCTURE SYSTEMS

François E. Cellier
Dept. of Electr. & Comp. Engr.
The University of Arizona
Tucson, Arizona 85721
U.S.A.
Cellier@ECE.Arizona.Edu

Martin Otter
Inst. für Robotik & Systemdynamik
DLR Oberpfaffenhofen
D–82230 Wessling
Germany
DF43@Master.DF.OP.DLR.DE

Hilding Elmqvist
Dynasim AB
Research Park Ideon
S–223 70 Lund
Sweden
Elmqvist@Dynasim.SE

ABSTRACT

The problem of describing variable structure models in a compact, object–oriented fashion is revisited and analyzed from the perspective of bond graph modeling. Traditionally, bond graphs have always been used to describe continuous–time physical processes with a fixed structure. Yet, this paper shall demonstrate that bond graphs are equally suitable to describe variable structure models as fixed structure models. Moreover, a bond graph description of variable structure models can teach us a lot about the essential properties of variable structure models, properties that are not easily visible when other modeling approaches are taken. The paper discusses issues related to causality reassignment and conditional index changes as a consequence of switching in a physical system.

Keywords: Bond graphs, variable structure system, computational causality, conditional index change, switching, object–oriented modeling, Dymola.

INTRODUCTION

When the causality strokes were added to the formerly acausal bond graphs, the bond graph community cheered. Something fundamental had been accomplished. Finally, bond graphers were able to determine once and for all, whether currents flowing through a circuit element were causing a voltage drop, or whether a potential difference between the two ends of the circuit element made a current flow through it. Quickly, it was discovered that, whereas capacitors and inductors had a preferred causality, two types of resistors were needed: *voltage–drop–causers* and *current–flow–causers*.

When reading some early papers about TUTSIM, one of the first software tools available for simulating bond graph models, one may discover that the designers of the tool praised as an important strength of the software that the user had to assign the causality strokes by hand, since this forced him or her to think carefully about the correct physical behavior of each bond graph element, which in turn would enhance the probability that a physically correct model was derived. ENPORT, on the other hand, did not share this virtue, which made it more likely that a gullible user would formulate a model that was physically meaningless, which would yet be accepted and processed by the simulation software without proper error messages being generated (van Dixhoorn 1982).

As recently as 1993, about half a dozen bond graphers struggled with the fact that ideal switch elements aren't cooperative since they don't allow to determine fixed causalities in the model once and for all (Granda & Cellier 1993). Switch elements play havoc with the causality strokes in the model. Some researchers introduced some tricks to freeze the causalities in one way or other, usually by making the switch non–ideal (Asher 1993, Ducreux *et al.* 1993); others came up with split circuits that showed different causalities depending on the switch position (Strömberg *et al.* 1993, Broenink & Wijbrans 1993, Lorenz 1993); other, more philosophical, treatises simply declared ideal switches to be "non–physical" (in the sense of representing an abstraction of un–modeled fast time constants), since evidently, every physical element must allow the determination of its causality.

What would physics be without causality? Isn't causality responsible for keeping up the order in the world and for ensuring that chaos would not reign over this universe of ours?

THE MYTH OF PHYSICAL CAUSALITY

When the Romans presented us with the *codex iuris romanum*, they codified, for the first time in human history, in strong and clear terms their beliefs about law and order, about cause and effect, to protect victims from perpetrators, to separate unambiguously between the guilty and the innocent. This Roman heritage has influenced our daily lives to this day more deeply and more profoundly than we may commonly think. It gives us immense personal satisfaction to be able to separate cause from effect, to distinguish between

the perpetrator and the victim, to ensure that the guilty is punished and the innocent is redeemed.

It is this deep–routed moral conviction that made us declare victory when causality strokes were introduced into the bond graph methodology. Finally, the bond graphers had learned to put everything in perspective, to cope with the realities of physics that distinguish clearly between *actio* and *reactio*.

Unfortunately, our ancestors were better at law than at physics. It is correct that, when two bodies are in contact with each other, each one experiences the same force from the other with opposite direction, such that the overall effect towards the outside is nil. Yet, the naming of this principle is most unfortunate, since it insinuates the distinction between a perpetrator and a victim. This however is a very human *moral* concept, and not a *physical* one. There is no physical experiment in the world that would allow us to distinguish between these two forces, i.e., to determine which is the *actio* and which is the *reactio*.

There could be such a thing as causality in physics. If I drive my car around and a tube breaks, it is a consequence that my car will lose its cooling water. Without enough coolant, it is a consequence that the engine will overheat. With an overheated engine, it is a consequence that a warning light will come on to alert the driver. If this warning light is ignored sufficiently long, it is a consequence that the engine will catch fire, and the car will burn.

Yet, this is quite different from the above. None of these statements is reciprocal. It is incorrect to write that the tube breaks because the car loses its cooling water. In all these cases, a discrete event takes place that somehow changes the structure of the model, as a consequence of which the system behavior changes. Usually, there is some time lapse between cause and effect, and "physical causality" could be defined to mean this type of relationship between causes and their effects.

If such a situation is modeled in a bond graph, some sort of ideal switching element will have to be introduced to denote the discrete event, as proposed e.g. in (Broenink and Wijbrans 1993, Lorenz 1993), and this is, as we already know, precisely the situation where the location of the causality strokes depends on the switch position, i.e., where the notion of a fixed causality breaks down. Hence, the term "causality," as it is being used by bond graphers around the globe, means something different. It means something much more closely related to the action/reaction principle. There is no physical experiment in the world that can be applied to a resistor to determine whether it is a "voltage–drop–causer" or a "current–flow–causer."

Bond graph causality was defined to mean the computational causality that is needed when a model of a physical system is to be expressed in state–space form, but physics doesn't know anything about state–space models and/or Runge–Kutta algorithms, physics only cares about such things as energy conservation and mass conservation. Hence, bond graph causality is not a physical principle at all. It is only an artifact of our traditional way of numerically simulating differential equation models, and if we were to employ a DAE solver instead of an ODE solver, the concept would become totally meaningless.

Thus, it is claimed that the introduction of causality strokes was *not* a victory at all. It was a big setback that hampered the further development of the bond graph methodology to this day.

THE IDEAL SWITCH ELEMENT

Using the notation of Dymola, an object–oriented modeling language for dynamic systems (Elmqvist 1978, Cellier 1991, Dynasim 1994), it is possible to describe the ideal switch element as follows:

```
model class Sw
  main cut A(e/f)
  terminal OpenSwitch
    0 = if OpenSwitch then f else e
end
```

A graphical representation of the switch element is shown in Fig.1. The Sw–element is a one–port element. The port (in Dymola called *cut*) is called A, and it contains two variables, the effort e (an *across* variable), and the flow f (a *through* variable). The Sw–element also references an information path (sometimes referred to as "activated bond" in the bond graph literature), called *OpenSwitch*. In Dymola, a *cut* that contains a single variable is called a *terminal*.

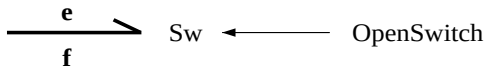

Figure 1: Bond graph representation of Sw–element.

The switch equation states that, at all times, either f or e are zero, depending on whether the switch is open (*OpenSwitch=true*) or closed (*OpenSwitch=false*), respectively. Dymola is able to cope with such a variable structure equation as shown in (Elmqvist *et al.* 1993).

If the switch is open ($f = 0$), the Sw–model computes the flow, and the causality stroke should be at the Sw–element. If the switch is closed ($e = 0$), the Sw–model computes the effort, and the causality stroke should be away from the Sw–element. Thus, the causality stroke indeed changes its position depending on the current value of the *OpenSwitch* variable. Thus, we are confronted with a *conditional causality reassignment* problem.

To understand a little better how the switch works, let us analyze a slightly modified switch model:

model class $Sw2$
  main cut $A(e/f)$
  terminal $OpenSw$
    $0 = OpenSw * f + (1 - OpenSw) * e$
end

Here, $OpenSw$ is a real–valued variable that assumes the values 1.0 (switch is open) and 0.0 (switch is closed). If the switch equation is solved for f (flow–causality):

$$f = \frac{OpenSw - 1}{OpenSw} \cdot e \qquad (1)$$

a division by zero will result as soon as the switch closes. Thus, the model of Eq.(1) is only valid as long as the switch is open. On the other hand, if the switch equation is solved for e (effort–causality):

$$e = \frac{OpenSw}{OpenSw - 1} \cdot f \qquad (2)$$

a division by zero will result as soon as the switch opens. Evidently, the model of Eq.(2) is only valid as long as the switch is closed. Thus, neither of the two models will work in both switch positions.

Until recently, it was thought that this problem is unsolvable, i.e., that two separate models would be needed at run time, each valid for one of the two switch positions only, between which the simulation program would have to toggle whenever switching takes place in the circuit. This is, however, an unfortunate proposition since a circuit containing 10 switch elements would call for $2^{10} = 1024$ different models.

A solution to this dilemma was finally proposed in (Elmqvist *et al.* 1993). A single model that works in both switch positions can easily be derived if and only if both causalities are compatible with the remainder of the bond graph, i.e., if changing the causality at the Sw–element does not force any incorrect causality on sources or undesired causality on storage elements. This evidently means that the switch equation *must* end up in an algebraic loop, since only such a model structure will allow a variable causality of the switch equation. The variable causality leads to different zero/non–zero patterns of the systems of equations of the algebraic loop for the different switch positions.

This concept will be demonstrated now by means of a simple example. An ideal electrical diode is a switch with internal modulation, i.e., the information variable is driven by a signal inside the switch itself:

model class $(Sw)\ D$
  new$(OpenSwitch)$ = **not** $e > 0$ **and not** $f > 0$
end

or using the algebraic variant of the switch:

model class $(Sw2)\ D2$
  new$(OpenSw)$ = **if not** $e > 0$ **and not** $f > 0$ **then** 1.0
                else 0.0
end

The diode model inherits all the variables and equations from the corresponding switch model, adding one additional equation to the set.

Figure 2 shows a half–wave rectifier circuit, and Fig.3 shows a bond graph representation of the same circuit. Evidently, both switch causalities are compatible with the remainder of the circuit.

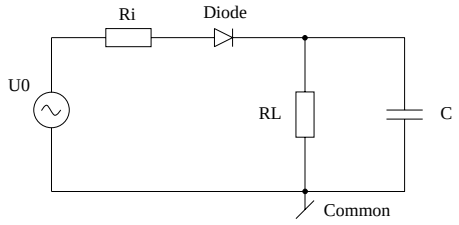

Figure 2: Half–wave rectifier circuit.

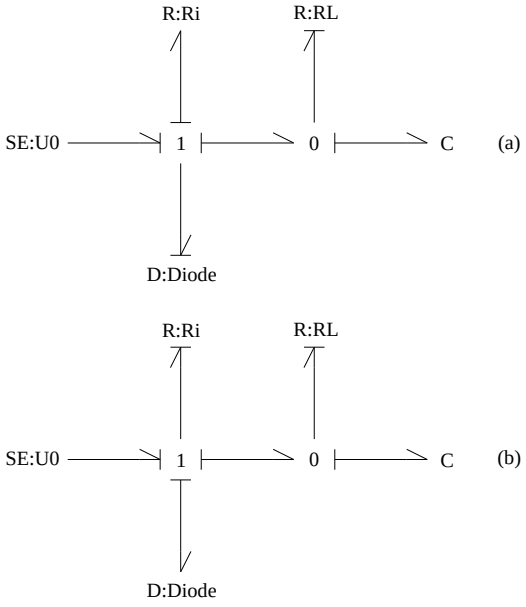

Figure 3: Rectifier bond graph with switch: (a) open, (b) closed.

Since Dymola doesn't currently understand the concept of a 1–junction yet (Dymola is a general–purpose object–oriented modeling language that had not originally been designed to deal with bond graph models), *bonds* are modeled in Dymola as symplectic gyrators:

model class *bond*
  cut $A(e/f),\ B(f/\text{-}e)$
  main cut $C[A, B]$
  main path $P < A - B >$
end

and all circuit elements must be attached to 0–junctions. A detailed description of how Dymola can be used to model bond graphs can be found in (Cellier 1991, Cellier 1992).

With these additional rules, it is now possible to describe the bond graph model of the half–wave rectifier circuit:

```
model Rectifier
  submodel (SE)   U0
  submodel (R)    Ri(R = 10), RL(R = 50)
  submodel (C)    C(C = 0.001)
  submodel (D)    Diode
  submodel (bond) B1, B2, B3, B4

  node      nSE, nRi, nD, n1, n0

  output    y
  local     w

  constant  pi = 3.14159
  parameter f = 50
    connect U0    at   nSE,
            B1    from nSE to n1,
            B2    from n1  to nRi,
            Ri    at   nRi,
            B3    from n1  to nD,
            Diode at   nD,
            B4    from n1  to n0,
            RL    at   n0,
            C     at   n0

  U0.E0 = sin(2 * 3.14159 * f * Time)
  y     = C.e
end
```

Once this model is entered into Dymola, Dymola adds additional equations that result from the topological connections (Kirchhoff's laws). The set of *sorted equations* for this problem looks as follows:

```
  [RectBond.w] = 2 * RectBond.pi * RectBond.f
  [B1.e] = sin(RectBond.w * RectBond.Time)

| C.e + [Diode.e] + Ri.e = B1.e
| 0 = if Diode.OpenSwitch then [B4.e] else Diode.e
| [Ri.e] = Ri.R * B4.e

  C.e = RL.R * [RL.f]
  RL.f + [C.f] = B4.e
  C.f = C.C * [C.dere]
  [Diode.newOpenSwitch] = not (Diode.e > 0)
                          and not (B4.e > 0)
```

In order to arrive at assignment statements that can be evaluated in sequence, the sorted equations must be solved for the variables enclosed in brackets, and algebraic loops, marked by "|" in the code, must be solved for the unknown loop variables. In the above example, the first two equations can be evaluated directly. Subsequently, an algebraic loop with three equations in three loop variables is present, indeed containing the switch equation. Finally, the last four equations can again be evaluated sequentially, once the algebraic loop has been solved.

The *solved equations* for this problem are:

```
SORTED AND SOLVED EQUATIONS
  RectBond.w = 2 * RectBond.pi * RectBond.f
  B1.e = sin(RectBond.w * RectBond.Time)

SYSTEM OF 3 SIMULTANEOUS EQUATIONS
  Q101 = if Diode.OpenSwitch then 1 else 0
  Q102 = if Diode.OpenSwitch then 0 else 1
  Q103 = Q101 * (B1.e - C.e)
  Q104 = Q102 * Ri.R - Q101
  Diode.e = Q103/Q104
  Q105 = Q102 * (B1.e - C.e)
  B4.e = Q105/Q104
  Q106 = Q102 * Ri.R * (B1.e - C.e)
  Ri.e = Q106/Q104
END OF SYSTEM OF SIMULTANEOUS EQUATIONS

  RL.f = C.e/RL.R
  C.f = B4.e - RL.f
  C.dere = C.f/C.C
  Diode.newOpenSwitch = not (Diode.e > 0)
                        and not (B4.e > 0)
END OF SORTED AND SOLVED EQUATIONS

ELIMINATED STATE DERIVATIVES AND OUTPUTS
  RectBond.y = C.e
```

It can be easily verified that the determinant $Q104$ is not equal to zero in either of the two switch positions. Thus, this is a single model valid for both switch positions. The price paid for this single model is that the equations describing the switch and the environment contaminated by it (i.e., those portions of the bond graph to which the changing causality propagates) are no longer those that can be read out from the bond graph directly. Instead in the process of compiling the model, an appropriate set of equations has been obtained through formula manipulation, more precisely by applying Cramer's rule to the set of algebraically coupled linear equations.

This ought to be reflected in the bond graph by eliminating the causal strokes altogether from those bonds that are associated with equations that appear in an algebraic loop.

Notice that the algebraic loop here is not an artifact. It is essential to the proper functioning of the ideal switch. An ideal switch equation that does not appear in an algebraic loop causes additional difficulties to the compiler, as will be shown in due course, and such a modeling element may indeed represent an abstraction of reality that is not compatible any longer with the laws of physics.

If an algebraic loop is large enough, it may be better to solve it numerically by generating assignment statements to the system matrix $\mathbf{A}$ and the right hand side vector $\mathbf{b}$, and by solving the resulting linear system of equations "$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$" by calls to, e.g., a LAPACK routine (Anderson *et al.* 1992). The presence of one or more ideal switch equations leads to rows in $\mathbf{A}$ that have exactly one non–zero element in every switch position. It is therefore possible to rearrange the rows and columns of $\mathbf{A}$ at run time during every structural change of a switch, such that

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{A}} \end{bmatrix}$$

where **I** is an identity matrix of dimension n_{sw}, corresponding to n_{sw} ideal switching equations. Consequently, the number of equations to be solved numerically is reduced. The described scheme has the practical advantage that a corresponding sorting procedure is already available in LAPACK (`SGEBAL`). `SGEBAL` is used in LAPACK for an optional first step before a matrix is being balanced and transformed to Schur–form for the purpose of determining its eigenvalues.

CONDITIONAL INDEX CHANGE DUE TO SWITCHING

Let us now look at another model. Figure 4 shows another simple circuit involving a diode, and Fig.5 shows a bond graph representation of the same circuit. Evidently, this time around, a causality change at the diode element causes the inductor to lose its preferred causality, i.e., the switch equation does not end up in an algebraic loop. The current through the inductor is a natural state variable, thus the current through the diode is computed in the inductor, and the switch equation *must* be solved for the voltage always. Evidently, this won't work.

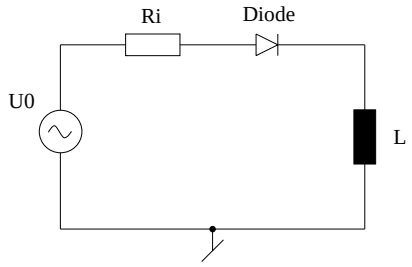

Figure 4: Inductive load circuit.

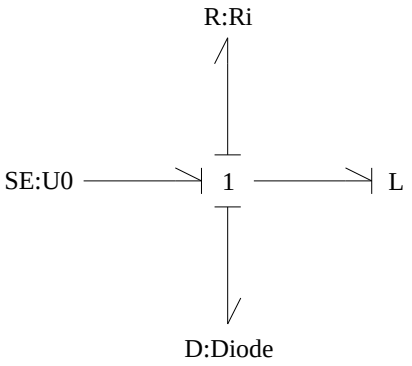

Figure 5: Inductive load bond graph.

What happens in this circuit is that, when the switch opens, the inductor is effectively taken out of the circuit. The model order changes from first order to zeroth order. Since the model still contains the inductor, the switching is experienced by the model as a conditional index change. The index of the DAE system describing this model jumps from zero to two (Brenan *et al.* 1989).[1]

A possible solution to this problem can be found by modifying the switch equation in the following way:

```
model class CSw
  main cut A(e/f)
  terminal OpenSwitch
    0 = if OpenSwitch then der(f) else e
    when OpenSwitch then
      init(f) = 0
    endwhen
end
```

Evidently, if the switch is open, $f = 0.0$, but this also implies that $\dot{f} = 0.0$. However, $\dot{f}$ is not a state variable, thus its causality will not be fixed, and thus, the switch equation will again appear inside an algebraic loop as it should. To avoid drift, f is initialized to 0.0 at the moment *when* the switch opens.

Using this modified switch model and a diode model inheriting these switch equations, the bond graph model can be encoded in Dymola using the same approach as in the previous example. The resulting DAE model is now of index two, since $Diode.f = L.f$ are two state variables. However, the model no longer exhibits a conditional index change. Thus, the Pantelides algorithm (Pantelides 1988, Cellier and Elmqvist 1993) can now be applied to reduce the index down to index one.

The resulting set of sorted equations is:

```
  [Ri.e] = Ri.R * Diode.f
  [RectBond.w] = 2 * RectBond.pi * RectBond.f
  [B1.e] = sin(RectBond.w * RectBond.Time)
| L.e + [Diode.e] + Ri.e = B1.e
| 0 = if Diode.OpenSwitch then [Diode.derf]
                           else Diode.e
| [L.e] = L.I * Diode.derf
  when Diode.OpenSwitch then
    [Diode.initf] = 0
  endwhen
  [Diode.newOpenSwitch] = not (Diode.e > 0)
                          and not (Diode.f > 0)
```

leading to the following set of *solved equations*:

[1] It is essential that the electrical switch opens when the current is zero, as this will automatically happen when a diode is used as the switch element. If this were not the case, an infinitely large voltage would temporarily have to appear across the inductor to pull the rest energy $0.5 \cdot L \cdot i^2$ out of the inductor within zero time. In practice, an arc would be drawn. Although it would be possible to handle such a situation in the model (replacing the Dirac impulse by modified initial conditions for the restart of the integration), this would complicate the matters considerably, and is therefore not considered here.

```
SORTED AND SOLVED EQUATIONS
  Ri.e = Ri.R * Diode.f
  RectBond.w = 2 * RectBond.pi * RectBond.f
  B1.e = sin(RectBond.w * RectBond.Time)

SYSTEM OF 3 SIMULTANEOUS EQUATIONS
  Q101 = if Diode.OpenSwitch then 1 else 0
  Q102 = if Diode.OpenSwitch then 0 else 1
  Q103 = Q101 * (B1.e - Ri.e)
  Q104 = Q102 * L.I - Q101
  Diode.e = -Q103/Q104
  Q105 = Q102 * (B1.e - Ri.e)
  Diode.derf = Q105/Q104
  Q106 = Q102 * L.I * (B1.e - Ri.e)
  L.e = Q106/Q104
END OF SYSTEM OF SIMULTANEOUS EQUATIONS

  when Diode.OpenSwitch then
    Diode.initf = 0
  endwhen
  Diode.newOpenSwitch = not (Diode.e > 0)
                        and not (Diode.f > 0)
END OF SORTED AND SOLVED EQUATIONS

ELIMINATED STATE DERIVATIVES AND OUTPUTS
  RectBond.y = Diode.f
```

As in the previous example, the determinant $Q104$ is different from zero in both switch positions, and consequently, the simulation will execute correctly.

Looking once more at the bond graph, it is *essential* that the causality of the diode changes at switching time. This is more important than the causality of the inductor. However, if the causality of the diode changes, then evidently, the causality of the inductor has to follow. Thus, both causality strokes should be eliminated. Looking at the *sorted equations*, it can be noted that indeed the inductor equation ends up inside the algebraic loop, as was to be expected from the previous discussion.

NON–IDEAL SWITCH ELEMENTS AND ARTIFICIAL STIFFNESS

Due to the difficulties associated with ideal switch elements (Elmqvist *et al.* 1994), many researchers prefer to replace the ideal switch element by a non–ideal switch element, in which the switch in its closed position is represented by a very small but not vanishingly small resistance, and in its open position by a very small but not vanishingly small conductance. In Dymola, such a model can be formulated as follows:

```
model class GSw
  main cut A(e/f)
  terminal OpenSwitch
  parameter GOpen = 0.0, RClosed = 0.0
    0 = if OpenSwitch then f - GOpen * e
                      else e - RClosed * f
end
```

which degenerates to the ideal switch when the default values of both *RClosed* and *GOpen* are in effect.

If the ideal switch has free causality, then so will the non–ideal switch. In this case, the behavior of the simulation models using ideal or non–ideal switches will be almost identical, except that the equations using non–ideal switches are a little more complicated. However, the systems of equations to be solved remain exactly the same with the only difference, that, for non–ideal switches, the zero/non–zero patterns of the system matrices do not change with switching. As a consequence, the number of operations to solve these systems of equations is the same for the ideal and for the non–ideal switch (assuming that no run–time sorting of the equations is provided), i.e., there will be no significant difference in execution speed.

On the other hand, if the ideal switch has fixed causality, then so will the non–ideal switch. Looking once more at the algebraic version of the (non–ideal) switch equation:

$$0 = OpenSw \cdot (f - GOpen \cdot e) + (1 - OpenSw) \cdot (e - RClosed \cdot f) \tag{3}$$

this equation can be solved for either f or e, leading to:

$$f = \frac{OpenSw \cdot (1 + GOpen) - 1}{OpenSw \cdot (1 + RClosed) - RClosed} \cdot e \tag{4}$$

$$e = \frac{OpenSw \cdot (1 + RClosed) - RClosed}{OpenSw \cdot (1 + GOpen) - 1} \cdot f \tag{5}$$

Whenever the ideal switch equation would have a zero denominator, the real switch equation will have a very small denominator, leading to artificially stiff system behavior.

From a physical perspective, when the switch is placed in series with an inductor, the inductor stores the energy $E_L = 0.5 \cdot L \cdot i^2$. When the switch opens, the remaining energy must be carried away rapidly by means of a large energy flow $P = dE/dt = u \cdot i$. Since the current through the inductor cannot jump, a large voltage has to appear to support the required large energy flow. Similarly for switches placed in parallel with capacitors. When the switch closes, the remaining energy stored in the capacitor $E_C = 0.5 \cdot C \cdot u^2$ must be removed quickly. This will require a very large current to flow temporarily.

From a mathematical point of view, it is *not* important whether a large energy flow actually takes place or not. In order to potentially support such a large energy flow, the Jacobian matrix of the ODE model must have an eigenvalue far out in the left–half complex plane, which makes the circuit invariably stiff.

Consequently, replacing an ideal switch by a non–ideal switch is relatively harmless precisely then when the ideal switch itself doesn't pose any problems, and leads to an artificially stiff system whenever the ideal switch would lead to a conditional index change in the system. This is quite unfortunate.

Although the trick with replacing f by $\dot{f}$ did work in the small example shown in this paper, it is unfortunate that the user needs to think about how to modify the switch equation, rather than being able to leave the necessary modifications to the system. This is cumbersome and in direct violation

of the request for object–orientation. After all, the physical system knows only one type of switch element, and it feels wrong that there should be a need to represent this element through different models depending on the environment in which it is being used. Until this problem is solved in a more satisfactory fashion, there is a need for non–ideal switching elements, and the user will grudgingly have to pay the price of potentially creating artificially stiff models.

CONCLUSIONS

It has been shown that the concept of causality, as propagated throughout the bond graph literature, is an oversold concept that had its justification at a time when bond graphs were drawn by hand onto sheets of paper to be translated manually into state–space models before feeding them to an ODE solver. However, bond graph causality does not represent a physical property, and its questionable use is limited to analyzing fixed structure models. The concept breaks down entirely when faced with variable structure models.

It was shown that powerful symbolic formula manipulation tools can be designed that allow the translation of variable structure models described by acausal bond graphs into properly executing state–space models in a fully automated fashion. In fact, such a tool already exists and has been used extensively in this paper to illustrate the new ideas and concepts.

A new class of interesting problems was discussed that exhibit conditional index changes in their DAE structures, a problem so far ignored by applied mathematicians working in the area of numerical DAE solvers. It is suggested that this class of problems be studied in the context of numerical DAE solvers, a research area that should lead to a further fruitful intensification of cooperations between the applied mathematics and engineering communities.

REFERENCES

Anderson, E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. 1992. *LAPACK User's Guide*, SIAM, Philadelphia.

Asher, G.M. 1993. "The Robust Modelling of Variable Topology Circuits Using Bond Graphs," *Proceedings ICBGM'93: International Conference on Bond Graph Modeling and Simulation* (San Diego, Calif., January 17–20), 126–131.

Brenan, K.E., S.L. Campbell, and L.R. Petzold. 1989. *Numerical Solution of Initial–Value Problems in Differential–Algebraic Equations*, North–Holland, New York.

Broenink, J.F., and C.J. Wijbrans. 1993. "Describing Discontinuities in Bond Graphs," *Proceedings ICBGM'93: International Conference on Bond Graph Modeling and Simulation* (San Diego, Calif., January 17–20), 120–125.

Cellier, F.E. 1991. *Continuous System Modeling*, Springer–Verlag, New York.

Cellier, F.E. 1992. "Hierarchical Non-Linear Bond Graphs: A Unified Methodology for Modeling Complex Physical Systems," *Simulation*, no. 58, (4): 230–248.

Cellier F.E. and H. Elmqvist. 1993. "Automated Formula Manipulation in Object–Oriented Continuous–System Modeling," *IEEE Control Systems*, no. 13 (2): 28–38.

Cellier, F.E., H. Elmqvist, M. Otter, and J.H. Taylor. 1993. "Guidelines for Modeling and Simulation of Hybrid Systems." In *Proc. IFAC World Congress* (Sydney, Australia, July 18–23), vol. 8, 391–397.

Ducreux, J.P., G. Dauphin–Tanguy, and C. Rombaut. 1993. "Bond–Graph Modelling of Commutation Phenomena in Power Electronic Circuits," *Proceedings ICBGM'93: International Conference on Bond Graph Modeling and Simulation* (San Diego, Calif., January 17–20), 132–136.

Dynasim. 1994. *Dymola — User's Manual*, Dynasim AB, Research Park Ideon, Lund, Sweden.

Elmqvist, H. 1978. *A Structured Model Language for Large Continuous Systems*, Ph.D. dissertation, Report CODEN: LUTFD2/(TFRT–1015), Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Elmqvist, H., F.E. Cellier, and M. Otter. 1993. "Object–Oriented Modeling of Hybrid Systems." In *Proc. ESS'93, European Simulation Symposium* (Delft, The Netherlands, October 25–28), xxxi–xli.

Elmqvist, H., F.E. Cellier, and M. Otter. 1994. "Object–Oriented Modeling of Power–Electronic Circuits Using Dymola." In *Proc. CISS'94, First Joint Conference of International Simulation Societies* (Zurich, Switzerland, August 22–25).

Granda, J.J., and F.E. Cellier, Eds. 1993. *Proceedings of ICBGM'93 — International Conference on Bond Graph Modeling and Simulation*, SCS Publishing, San Diego, Calif.

Lorenz, F. 1993. "Discontinuities in Bond Graphs: What Is Required?" *Proceedings ICBGM'93: International Conference on Bond Graph Modeling and Simulation* (San Diego, Calif., January 17–20), 137–142.

Pantelides, C.C. 1988. "The Consistent Initialization of Differential–Algebraic Systems," *SIAM J. Scientific and Statistical Computing*, no. 9 (2): 213–231.

Strömberg, J.-E., J. Top, and U. Söderman. 1993. "Variable Causality in Bond Graphs Caused by Discrete Effects," *Proceedings ICBGM'93: International Conference on Bond Graph Modeling and Simulation* (San Diego, Calif., January 17–20), 115–119.

van Dixhoorn, J.J. 1982. "Bond Graphs and the Challenge of a Unified Modelling Theory of Physical Systems." In *Progress in Modelling and Simulation*, F.E. Cellier, ed. Academic Press, London, 207–245.