Improvements in BondLib, the Modelica Bond Graph Library

Alberto de la Calle*, François E. Cellier†, Luis J. Yebra*, Sebastián Dormido‡
*Automatic Control Group, CIEMAT-Plataforma Solar de Almería, Tabernas, Spain
Email: {alberto.calle, luis.yebra}@psa.es
†Institute of Computational Science, ETH Zürich, Zürich, Switzerland
Email: fcellier@inf.ethz.ch
‡Dept. of Computer Science and Automatic Control, UNED, Madrid, Spain
Email: sdormido@dia.uned.es

***Abstract*—Bond graphs are a natural method for representing power flows in physical systems and they represent the most basic graphical paradigm that is still fully object-oriented. This paper describes a new version of BondLib, the Modelica bond graph library. The library was created for Modelica with graphical Dymola support, and its object-oriented design allows the wrapping of bond graph models offering a user-friendly environment for modeling physical system in multiple energy domains. The new improvements try to reduce the deployment and training cost for end-users by making the error diagnostics easier.**

***Keywords*—Modelica; bond graph; energy modeling; balanced models; error diagnostic;**

I. Introduction to Bond Graphs

The conservation of energy is a common feature in every physical systems. When modeling a physical system, the modeler has to ensure that this conservation principle is never violated. Bond graphs make this check a trivial task because it is implicit in the formalism [1].

A bond graph is a graphical representation of a physical system. This representation provides bi-directional information of the exchanged energy between different elements of the graph.

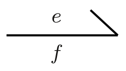


Fig. 1. Representation of a bond

A bond depicts the power flow between two elements of a physical system. It is represented by a harpoon and carries two adjugate variables: an effort, e, which is an extensive variable, and a flow, f, which is an intensive variable. The product of both is defined to be power, $P = e \cdot f$.

Efforts and flows can represent different physical properties but bond graphs have, in principle, a neutral modeling domain. The assignment of two physical variables to an effort and a flow determines its domain. A bond graph can incorporate multiple domains seamlessly for representing systems that operate in multiple energy domains. Beside e and f, there are two additional physical quantities that play an important role in the bond graph methodology: the generalized momentum, p, and the generalized position, p. They are defined as the time integrals of e and f, respectively. The most common assignments of domains are shown in Table I.

TABLE I. Usual domain assignments in bond graph methodology

Domain	Effort	Flow	Generalized momentum	Generalized position
Electrical	Voltage	Current	Magnetic flux	Charge
Translational mechanics	Force	Velocity	Momentum	Position
Rotational mechanics	Torque	Angular velocity	Torsion	Angle
Hydraulics	Pressure	Volumetric flow	Pressure momentum	Volume
Thermodynamics	Temperature	Entropy flow	-	Entropy
Chemical	Chemical potential	Molar flow	-	Number of moles

TABLE II. Basic modeling elements of bond graph methodology

Name	Code	Equation
0-junction	**0**	$e_i = e_{i+1}$ $\text{sum}(f) = 0$
1-junction	**1**	$f_i = f_{i+1}$ $\text{sum}(e) = 0$
Resistance	**R**	$e = R(f)$
Capacitance	**C**	$e = C(q)$ $\text{der}(q) = f$
Inductance	**I**	$f = I(p)$ $\text{der}(p) = e$
Source of effort	**Se**	$e = e_0$
Source of flow	**Sf**	$f = f_0$
Transformer	**TF**	$e_1 = m \cdot e_2$ $f_2 = m \cdot f_1$
Gyrator	**GY**	$e_1 = m \cdot f_2$ $e_2 = m \cdot f_1$

The modeler can subdivide every physical system into small components such that each component exhibits a specific energy behavior: they can generate, store, transport, transform, dissipate (convert to heat) and delete energy. In bond graph terminology, sources and sinks of energy are the source of effort (Se) or a source of flow (Sf) elements, dissipation is modeled using resistance (R), whereas the capacitance (C) and

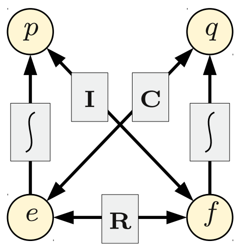


Fig. 2. Relations between bond graph variables

the inductance (I) are used for energy storage. The relations between the bondgraphic variables e, f, p and q determine the kind of element that we are dealing with (cf. Fig. 2).

The power flows between elements can be distributed with two dual junctions, 0-junctions and 1-junctions. In a 0-junction, the efforts are set equal whereas the flows add up to zero. Correspondingly in a 1-junction, the flows are set equal, whereas the efforts add up to zero. These junctions are the nodes of the model and determine if the elements in the graph are getting connected in series or in parallel.

In closing, there are two more basic elements in bond graph methodology; the transformer (TF) and the gyrator (GY). Both are used in the transformation of energy, either from one domain to another, or within a domain. Bond graph puritans argue that these elements (v. Table II) exhaust all of physics and no other elements should be offered.

II. Introduction to BondLib, the Modelica bond graph library

BondLib [2] was created by F. E. Cellier and his students with the aim of modeling bond graphs through the graphical user interface of Modelica tools. It was presented in Hamburg-Harburg in the framework of a Modelica Conference (2005) winning the first award for a free Modelica library. Since its release, the library has been updated several times to include new developed models and new wrapped sub-libraries.

In spite of bond graphs being capable of modeling all types of physical systems, there rarely are the best methodology to do so. The bond graph elements are the the most primitive graphical components that are still fully object-oriented; therefore when modeling a complex physical system in this way, the resulting bond graph would be unnecessarily large and difficult to read. The efficacy of this methodology lies in taking advantage of the object-oriented paradigm and wrapping bond graph models at higher abstraction levels. In this way, end-users can have a domain specific interface with elements that look familiar to them, but the bottom graphical layer is based on bond graph technology and is therefore easily maintainable.

The library has been divided in several sub-libraries that wrap each physical domain included. For modeling electronic analog circuits, the library offers a partial re-implementation of the standard analog electrical Modelica library using bond graph technology. In addition, a complete implementation of Spice built on bond graphs has been included [3]. Translational or rotational 1D mechanical systems can be modeled with the corresponding two sub-libraries. These sub-libraries are analogous to the standard 1D mechanical Modelica library, but due to the underlying bond graph technology, the connectors of both libraries are incompatible with those of the standard library. The thermal domain without convective heat flows has been wrapped in a sub-library that re-implements the standard thermal Modelica library. A sub-library for modeling and simulating mass and information flows of continuous-time systems with the System Dynamics methodology is also available in BondLib. Hydraulic and pneumatic sub-libraries wrap the hydraulic and pneumatic physical domains and they have been added in the new library version.

Two more libraries related with BondLib have been developed. MultiBondLib [4] offers several wrapped sub-libraries for higher-level descriptions of 2D and 3D mechanical systems. ThermoBondLib offers graphical support for modeling convective flows [5] and chemical reactions [6].

III. BondLib 2.4

The principal concepts of the previous version of BondLib (2.4) are described in this section.

A. Bonds

As explained in §I, each bond carries two variables, e and f, therefore the bondgraphic connectors need to carry at least these two variables. Another (non-physical) variable that encodes the direction of the bonds is necessary for processing the bondgraphic junctions. This variable, d, indicates the direction of positive power flow. It is encoded $d = -1$ at the emanating bondgraphic connector and $d = +1$ at the receiving. The bondgraphic connector is defined as follows, where all variables are of the potential type:

```
connector BondCon "Bi-directional bondgraphic connector"
  Real e "Bondgraphic effort variable";
  Real f "Bondgraphic flow variable";
  Real d "Directional variable";
end BondCon;
```

Listing 1. A-causal bondgraphic connector v2.4

The bond is defined as shown in List. 2.

```
model Bond
  "The a-causal bond model of the bond graph library"
  Interfaces.BondCon BondCon1 "Left bond graph connector";
  Interfaces.BondCon BondCon2 "Right bond graph connector";
equation
  BondCon2.e = BondCon1.e;
  BondCon2.f = BondCon1.f;
  BondCon1.d = -1;
  BondCon2.d = +1;
end Bond;
```

Listing 2. A-causal bond v2.4

Since bonds carry two physical variables, e and f, it is necessary to generate two equations to compute their values. This means that, in every case, one of the two variables is computed at each end of the bond. Modelica is capable of deciding the computational causality of all equations [7], but often, the causality of the bondgraphic elements are pre-defined for improved readability of the bond graph. To this end, the causal bond side, where the flow variable is computed, is marked with a vertical stroke (v. Fig. 3) and their connectors have an input/output prefix for each variable. Causal and a-causal bonds can be arbitrarily mixed, but it is recommended to use causal bonds as much as possible to better identify modeling errors.

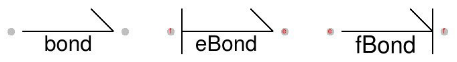


Fig. 3. Graphical representation of BondLib bonds

B. Junctions

In BondLib, the junctions, just like other bondgraphic elements, are defined using two levels. The lower level, called interface, is encoded as a partial model and defines the efforts and flows taking incount the directional d variable. The upper level, the element model, inherits the interface and defines the element equations.

For example, the 1-junction with three bond connectors is defined as:

```
model J1p3 "Model of ThreePort 1-junction"
  extends Interfaces.ThreePortOne;
equation
  f[2:3] = f[1:2];
  sum(e) = 0;
end J1p3;
```

Listing 3. ThreePort 1-junction model v2.4

```
partial model ThreePortOne
  "Partial model invoking three bondgraphic connectors"
  Real e[3] "Bondgraphic effort vector";
  Real f[3] "Bondgraphic flow vector";
  Interfaces.BondCon BondCon1 "First bond graph connector";
  Interfaces.BondCon BondCon2 "Second bond graph connector";
  Interfaces.BondCon BondCon3 "Third bond graph connector";
equation
  e[1] = BondCon1.d*BondCon1.e;
  f[1] = BondCon1.f;
  e[2] = BondCon2.d*BondCon2.e;
  f[2] = BondCon2.f;
  e[3] = BondCon3.d*BondCon3.e;
  f[3] = BondCon3.f;
end ThreePortOne;
```

Listing 4. ThreePortOne interface v2.4

C. Basic elements

All elements in Table II are encoded with four basic interfaces that depend on the number of connections (one or two port) and the power flow direction (active or passive).

For example, the linear capacitor (one port and passive) is defined as:

```
model C "The bondgraphic linear capacitor element"
  extends Interfaces.PassiveOnePort;
  parameter Real C=1 "Bondgraphic Capacitance";
equation
  f = C*der(e);
end C;
```

Listing 5. Linear capacitor model v2.4

```
partial model PassiveOnePort
  "Partial model invoking one bondgraphic connector"
  Real e "Bondgraphic effort";
  Real f "Bondgraphic flow";
  Interfaces.BondCon BondCon1 "Bond graph connector";
equation
  e = BondCon1.e;
  f = BondCon1.d*BondCon1.f;
end PassiveOnePort;
```

Listing 6. PassiveOnePort interface v2.4

The linear capacitor listing reveals why some elements have a fixed causality, because, in this case, it is obvious that this element calculates the effort and uses the flow as an input variable.

D. Wrapped elements

Since bond graphs offer a low-level interface, it is always possible to wrap bond graphs inside other models. The relevance of this technique is explained in §II. For this purpose, two wrappers per domain are defined and used to transform physical connectors to bondgraphic connectors.

Depending on the domain assignment, there are two kinds of wrappers: the *transformer* wrappers whose behavior is like a transformer (the bondgraphic f variable is the conventional *flow* variable in the domain, e.g. the electrical domain) or the *gyrator* wrappers whose behavior is like a gyrator (the bondgraphic e variable is chosen as the conventional *flow* variable in the domain, e.g. the translational mechanics domain).

For example in the electrical domain, the wrappers are:

```
model El2BG "Electrical to bond graph conversion"
  Modelica.Electrical.Analog.Interfaces.PositivePin p
    "Electrical connector";
  BondLib.Interfaces.BondCon BondCon1
    "Bond graph connector";
equation
  BondCon1.e = p.v;
  BondCon1.f = p.i;
end El2BG;
```

Listing 7. El2BG interface v2.4

```
model BG2El "Bond graph to electrical conversion"
  Modelica.Electrical.Analog.Interfaces.NegativePin n
    "Electrical connector";
  BondLib.Interfaces.BondCon BondCon1
    "Bond graph connector";
equation
  BondCon1.e = n.v;
  BondCon1.f = -n.i;
end BG2El;
```

Listing 8. BG2El interface v2.4

Using these wrappers, the electrical linear resistor model (v. Fig. 4) can be wrapped employing only the graphical interface (v. Fig. 5) in an easy way.

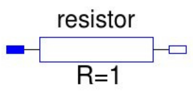


Fig. 4. BondLib electrical linear resistor

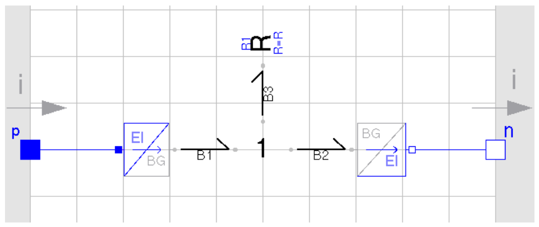


Fig. 5. Graphical wrapping of BondLib electrical linear resistor

IV. BondLib 3.0

Despite the numerous advantages of wrapping, the modeler does sometimes not have choice and develops a model of a complex physical system in just one bond graph, e.g. [8]. In that kind of a situation, the resulting model contains a big number of connections and elements, and therefore, one simple error, such as a bad connection or a structurally singular element, can cost the modeler hours in debugging his code, because the only help he receives it is an error message like *"The problem is structurally singular: It has 1103 scalar unknowns and 1102 scalar equations..."*. At other times, the modeler may have used model wrapping, but the code contains a simple error in a lower-level model, and the search of that error can again be tedious because the error message provides little information of where it is.

The problem is that Modelica is an a-causal language and in contrast to causal languages, where the use of input/output blocks make it easy to verify locally that all inputs have been connected, this local verification is not possible in Modelica. Structural analysis can only be performed globally after the model has already been flattened. Olsson et al. [9] have developed a methodology to provide diagnostics earlier in Modelica 3.0 or later. Their approach, however, demands a series of restrictions when a model is being developed. The main restriction (in normal cases) is that *"all non-partial model or block classes must be locally balanced"*, i.e., the number of equations in the top-level model = number of unknowns - number of inputs - number of flow variables. As a consequence, for each flow variable in a connector there must exist an identical number of non-causal non-flow variables. The new version of the BondLib library imposes the restrictions of [9] in all its components with the aim of making error messages easier for the end-users to interpret.

A. Bonds

The old bondgraphic connectors (v. §III-A) exceeded the number of non-causal variables of the imposed new restrictions. Therefore, f has now been chosen as a flow variable and e as non-causal potential variable. Although the new f variable carries the information of the power flow direction, this is unfortunately not enough information for the junctions to be correctly processed; therefore, a d variable is still needed. This, cannot be a non-causal variable in the connector, and consequently, it needs an input/ouput prefix to determine its causality.

Two bondgraphic a-causal connectors have been developed. BondCon_a is the male *bond* connector that must be used only by the bonds. It computes the value of the d variables on both ends of the bond. The remaining components in BondLib must all use BondCon_b, the female *element* connector that receives the d information from the connecting bond. In this way, BondCon_a (full circle) connectors are always connected to BondCon_b (empty circle) connectors. As an immediate benefit, it is impossible in BondLib 3.0 to connect two bonds in series or to attach an element at a junction without placing a bond in between.

```
connector BondCon_a "Bond connector"
  Real e "Bondgraphic effort variable";
  flow Real f "Bondgraphic flow variable";
  output Real d "Directional variable";
end BondCon_a;
```

Listing 9. A-causal bond connector v3.0

```
connector BondCon_b "Element connector"
  Real e "Bondgraphic effort variable";
  flow Real f "Bondgraphic flow variable";
  input Real d "Directional variable";
end BondCon_b;
```

Listing 10. A-causal element connector v3.0

The a-causal bond is defined as follows, where the two bondgraphic connectors of the past have been replaced by two BondCon_a connectors.

```
model Bond
  "The a-causal bond model of the bond graph library"
  Interfaces.BondCon_a BondCon1 "Left bond graph connector";
  Interfaces.BondCon_a BondCon2 "Right bond graph connector"
      ;
equation
  BondCon2.e = BondCon1.e;
  BondCon2.f = BondCon1.f;
  BondCon1.d = -1;
  BondCon2.d = +1;
end Bond;
```

Listing 11. A-causal bond v3.0

As in §III-A, the causal bonds are composed with connectors with an input/output prefix for each variable.

B. Junctions

Taking advantage of the object-oriented paradigm, only the interface level had to be modified in the new BondLib library. BondCon connectors have been replaced by the BondCon_b connectors in all bondgraphic elements except for the bonds themselves. The d variable is conveniently used with the e and f internal variables to obtain the 0-junction and 1-junction. List. 12 and List. 13 show two examples of the new interface.

```
partial model ThreePortZero
  "Partial model invoking three bondgraphic connectors"
  Real e[3] "Bondgraphic effort vector";
  Real f[3] "Bondgraphic flow vector";
  Interfaces.BondCon_b BondCon1
    "First bond graph connector";
  Interfaces.BondCon_b BondCon2
    "Second bond graph connector";
  Interfaces.BondCon_b BondCon3
    "Third bond graph connector";
equation
  e[1] = BondCon1.e;
  f[1] = BondCon1.f;
  e[2] = BondCon2.e;
  f[2] = BondCon2.f;
  e[3] = BondCon3.e;
  f[3] = BondCon3.f;
end ThreePortZero;
```

Listing 12. ThreePortZero interface v3.0

```
partial model ThreePortOne
  "Partial model invoking three bondgraphic connectors"
  Real e[3] "Bondgraphic effort vector";
  Real f[3] "Bondgraphic flow vector";
  Interfaces.BondCon_b BondCon1
    "First bond graph connector";
  Interfaces.BondCon_b BondCon2
    "Second bond graph connector";
  Interfaces.BondCon_b BondCon3
    "Third bond graph connector";
equation
  e[1] = BondCon1.d*BondCon1.e;
  f[1] = BondCon1.d*BondCon1.f;
  e[2] = BondCon2.d*BondCon2.e;
  f[2] = BondCon2.d*BondCon2.f;
  e[3] = BondCon3.d*BondCon3.e;
  f[3] = BondCon3.d*BondCon3.f;
end ThreePortOne;
```

Listing 13. ThreePortOne interface v3.0

C. Basic elements

As stated in §IV-B, only the interfaces have been modified for upgrading the bondgraphic elements. All higher-level models remain unchanged. The idea is that if one user has developed his own elements using the old interfaces, these models are going to be updated without effort when the BondLib is being updated to the new version.

Although passive and active one-port interfaces could be combined to just one interface with the new connectors, this option was rejected in order to guarantee that the direction of the bond does not affect the global graph. However, the two-port interfaces have been combined to just one because, in this case, an error in the bond direction is not possible.

D. Wrapped elements

Once the bondgraphic elements have been updated using the basic interfaces, only the wrappers still need to be updated. Unfortunately, this update is not a trivial task. *Transformer* wrappers cannot have a 0-junction behavior, as someone might think, because some components in the BondLib sub-libraries do not have symmetrical behavior. In order to preserve models developed by users without modifications, the authors have decided to continue to allow models with non-symmetric behavior to exist and update only the wrappers. Examples of this are the wrappers in the electrical domain.

```
model El2BG "Electrical to bond graph conversion"
  Modelica.Electrical.Analog.Interfaces.PositivePin p
    "Electrical connector";
  Interfaces.BondCon_b BondCon1 "Bond graph connector";
equation
  BondCon1.e = p.v;
  BondCon1.f*BondCon1.d = p.i;
end El2BG;
```

Listing 14. El2BG interface v3.0

```
model BG2El "Bond graph to electrical conversion"
  Modelica.Electrical.Analog.Interfaces.NegativePin n
    "Electrical connector";
  Interfaces.BondCon_b BondCon1 "Bond graph connector";
equation
  BondCon1.e = n.v;
  BondCon1.f*BondCon1.d = -n.i;
end BG2El;
```

Listing 15. BG2El interface v3.0

Gyrator wrappers do not have the same problem, because here, a lack of symmetry does not make they behave like a 0-junction.

V. Error diagnostics

The restrictions described in [9] help users in error diagnostics. The use of balanced connectors in complex models guarantees an efficient search of sub-models with structural singularities at a local rather than global level. Also, as a consequence of using two different connectors in BondLib 3.0, users obtain one of the most important advantages: the fixed causality in the d variable forces users to connect each bond to an element thereby eliminating an important source of connection errors right from the start.

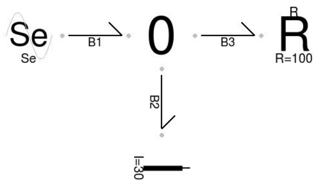
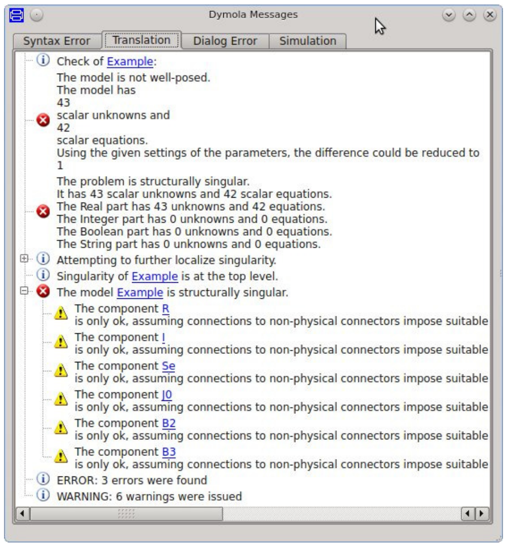


Fig. 6. Diagnostic example



```
Check of Example:
The model is not well-posed.
The model has
43
scalar unknowns and
42
scalar equations.
Using the given settings of the parameters, the difference could be reduced to
1
The problem is structurally singular.
It has 43 scalar unknowns and 42 scalar equations.
The Real part has 43 unknowns and 42 equations.
The Integer part has 0 unknowns and 0 equations.
The Boolean part has 0 unknowns and 0 equations.
The String part has 0 unknowns and 0 equations.
Attempting to further localize singularity.
Singularity of Example is at the top level.
The model Example is structurally singular.
  The component R
  is only ok, assuming connections to non-physical connectors impose suitable
  The component I
  is only ok, assuming connections to non-physical connectors impose suitable
  The component Se
  is only ok, assuming connections to non-physical connectors impose suitable
  The component J0
  is only ok, assuming connections to non-physical connectors impose suitable
  The component B2
  is only ok, assuming connections to non-physical connectors impose suitable
  The component B3
  is only ok, assuming connections to non-physical connectors impose suitable
ERROR: 3 errors were found
WARNING: 6 warnings were issued
```

Fig. 7. First test Dymola message with BondLib v.2.4

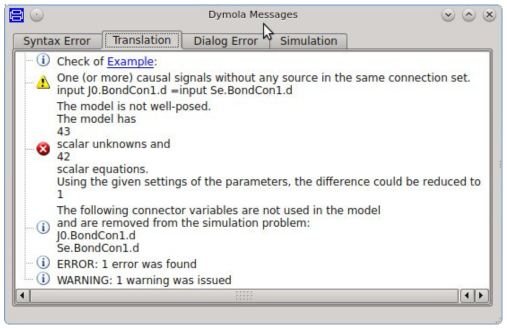
In order to demonstrate the different error reporting of Dymola under certain errors with BondLib 2.4 and 3.0, a series of located errors were tested using both libraries. The model used in several diagnostic tests involves a simple RL circuit (v. Fig. 6).

A. First test: Connecting two elements without a bond in between

The diagnostic example was modified suppressing *B1* and connecting *Se* to *0-junction* without a bond. After a model check, Dymola showed different error messages depending on the library version (v. Fig. 7 and Fig. 8). Both versions detected the error, but the model that used the new BondLib version pointed out in the warning dialog where exactly the error was. Two d variables were left without a value and were consequently removed from the model.

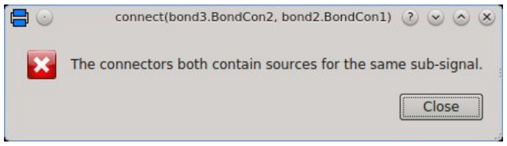
B. Second test: Connecting two bonds together

An additional bond between *Se* and *B1* was introduced in the second diagnostic example. The model that used the 2.4 version showed a poorly readable error message similar to Fig. 7. Using BondLib 3.0, users cannot make this mistake any longer because Dymola shows an error message box (v. Fig. 9) as soon as the user tries to connect two bonds. Dymola does not let users connect two output variables to each other.



```
Check of Example:
One (or more) causal signals without any source in the same connection set.
input J0.BondCon1.d =input Se.BondCon1.d
The model is not well-posed.
The model has
43
scalar unknowns and
42
scalar equations.
Using the given settings of the parameters, the difference could be reduced to
1
The following connector variables are not used in the model
and are removed from the simulation problem:
J0.BondCon1.d
Se.BondCon1.d
ERROR: 1 error was found
WARNING: 1 warning was issued
```

Fig. 8. First test Dymola message with BondLib v.3.0



```
connect(bond3.BondCon2, bond2.BondCon1)
The connectors both contain sources for the same sub-signal.
```

Fig. 9. Bonds connection error in BondLib v3.0

C. Third test: Submodel structural singularity

One equation in *R* was suppressed. Just like in the first test, Dymola detected the error with both BondLib versions. Dymola could not guide the user in locating the error in the model with BondLib 2.4. However, Dymola was able to point out the component where the error is (v. Fig. 10) when the model uses the new library version.

D. Fourth test: Global structural singularity

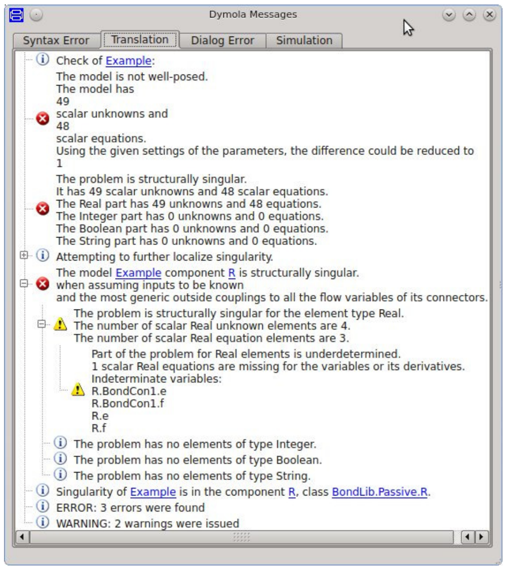
A structural singularity was added in the model but no element had it. *R* was replaced by a second *Se*, therefore *0-junction* is not able to compute the f and e variables. The models of both libraries produced poorly readable error messages, but, at least when using the new library version, Dymola indicated that the e variable is over-determined whereas the f variable is under-determined. Following that clue, users could find the origin of the error. A better solution would be to use causal bonds instead of a-causal bonds. In that case, the Dymola error message will be able to point at the under-determined f variable in B1 and B3.

E. Fifth test: Wrapping sub-models with structural singularity

In this test, with the aim of visualizing the Dymola reaction to an error in a wrapped model, one equation was suppressed in *BondLib.Thermal.RSth*, a model available in BondLib. Simulating the *TwoMasses* test example offered in BondLib, the tool detected errors with both libraries. Since thermal connectors observe the restrictions imposed by [9], the error was localized at *BondLib.Thermal.HeatTransfer.Passive.ThermalConductor* already when BondLib 2.4 was used. However, Dymola was able to point to the *RSth* model as the origin of the error when BondLib 3.0 was used.

VI. Conclusions

This paper introduces a new version of BondLib, the Modelica library for modeling with bond graphs. The new



```
Check of Example:
The model is not well-posed.
The model has
49
scalar unknowns and
48
scalar equations.
Using the given settings of the parameters, the difference could be reduced to
1
The problem is structurally singular.
It has 49 scalar unknowns and 48 scalar equations.
The Real part has 49 unknowns and 48 equations.
The Integer part has 0 unknowns and 0 equations.
The Boolean part has 0 unknowns and 0 equations.
The String part has 0 unknowns and 0 equations.
Attempting to further localize singularity.
The model Example component R is structurally singular.
when assuming inputs to be known
and the most generic outside couplings to all the flow variables of its connectors.
    The problem is structurally singular for the element type Real.
    The number of scalar Real unknown elements are 4.
    The number of scalar Real equation elements are 3.
        Part of the problem for Real elements is underdetermined.
        1 scalar Real equations are missing for the variables or its derivatives.
        Indeterminate variables:
        R.BondCon1.e
        R.BondCon1.f
        R.e
        R.f
    The problem has no elements of type Integer.
    The problem has no elements of type Boolean.
    The problem has no elements of type String.
Singularity of Example is in the component R, class BondLib.Passive.R.
ERROR: 3 errors were found
WARNING: 2 warnings were issued
```

Fig. 10. Third test Dymola message with BondLib v3.0

improvements are oriented to reduce the deployment and training cost for end-users making it easier for them to diagnose errors in their models. These improvements have been developed trying to preserve most of the existing code, limiting the modifications to the lower-most levels of the library architecture, thereby guaranteeing the reuse of models developed by the users with the latest library version. The changes in the code have been discussed in this article, and a series of diagnostic examples have been shown.

Acknowledgments

The authors would like to thank the CIEMAT research centre, Spanish Ministry of Economy and Competitiveness and ERDF funds for financing this work under the National Plan Project, POWER project, DPI2010-21589-C05-02.

References

[1] F. E. Cellier, *Continuous System Modeling*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, May 1991.

[2] F. E. Cellier and A. Nebot, "The Modelica bond graph library," in *Modelica Conference*, Hamburg, Germany, 2005, pp. 57–65.

[3] F. E. Cellier, C. Clauss, and A. Urquía, "Electronic circuit modeling and simulation in Modelica," in *EUROSIM*, Ljubljana, Slovenia, 2007, pp. 1–10.

[4] D. Zimmer and F. E. Cellier, "The Modelica Multi-Bond Graph Library," in *Modelica Conference*, Vienna, Austria, 2006, pp. 559–568.

[5] F. E. Cellier and J. Greifeneder, "ThermoBondLib–A New Modelica Library for Modeling Convective Flows," in *Modelica Conference*, Bielefeld, Germany, 2008, pp. 163–172.

[6] J. Greifeneder and F. E. Cellier, "Modeling Chemical Reactions Using Bond Graphs," in *Intl. Conf. on Bond Graph Modeling and Simulation*, Genoa, Italy, 2012, pp. 110–121.

[7] H. Elmqvist, "A Structured Model Language for Large Continuous Systems," Ph.D. dissertation, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1978.

[8] F. E. Cellier, A. Nebot, and J. Greifeneder, "Bond graph modeling of heat and humidity budgets of biosphere 2," *Environmental Modelling & Software*, vol. 21, no. 11, pp. 1598–1606, Nov. 2006.

[9] H. Olsson, M. Otter, S. E. Mattsson, and H. Elmqvist, "Balanced Models in Modelica 3.0 for Increased Model Quality," in *Modelica Conference*, Bielefeld, Germany, 2008, pp. 21–33.