Initialization of Equation-based Hybrid Models within OpenModelica

Lennart A. Ochel[1], Bernhard Bachmann[1]

[1]Department Mathematics and Engineering, University of Applied Sciences Bielefeld, Germany,
{lennart.ochel,bernhard.bachmann}@fh-bielefeld.de

Abstract

Modelica is a multi-domain object-oriented modeling language designed for time-dependent systems. The time-dependent part is usually described with "ordinary differential equations". In addition to that, it is possible to express algebraic and difference equations. As a result a Modelica model will be merged to a hybrid differential algebraic equation system.

The initialization process is prior to each simulation and must therefore be solved before any simulation can be started. Modelica provides high-level features to describe the initialization problem. This leads often into various problems. The initialization is usually a system-level issue. Therefore, high knowledge about the system is necessary.

In OpenModelica two major methods are implemented to solve the initialization problem. Both methods are totally different and are used for different initialization issues. Both methods will be discussed within this paper.

Keywords initialization, hybrid models, homotopy, start value, OpenModelica

1. Introduction

Primary linguistic constructs of Modelica to specify the initialization are initial equations and initial algorithms. It is possible that the initialization is not unique, even if initial conditions are fully specified. This means that the number of unknowns (in the case of initialization) is equal to the number of initial conditions. The non-uniqueness is caused by nonlinearities.

As a result, the modeler is not able to control the initialization completely, if just initial equations and initial algorithms are used. To retain control of the initialization, additional linguistic devices such as the homotopy operator and variable attributes (e.g. start values) are available in Modelica. Since homotopy is quite an advanced feature, the modeler may prefer the use of start values.

The influence of start values is in most implementations rather small. They are mostly used as an initial guess for nonlinear systems. Therefore, it is necessary to provide start values for variables, which are involved in these nonlinear systems. Moreover, it is important to know about dependencies of a given model and about suitable start values for just these variables.

This work will show how it is possible to increase the influence of start values during initialization a lot and to provide the modeler full control on the solution for his initialization problem. This is done by the first major method based on numerical algorithms and an extension called "Start Value Homotopy".

The second major approach is based on symbolical transformations. It creates a complete dependence graph for the initialization. Hence, the initial equations are sorted and transformed to a system that can be explicitly evaluated, except involved algebraic loops. This leads to a much faster and more accurate solution compared to the numeric approach.

2. Modelica Constructs for Initialization

Modelica contains several language constructs that influence the initialization (see [1]). These constructs can be categorized like follows:

Firstly, there are initial equations and initial algorithms that declare additional equations and algorithms to the time-dependent system. These special equations and algorithms are only active during initialization and are added to the simulation equations. In case of a hybrid model, when-equations are only considered, if activated using the `initial()` operator.

attribute	Real	Integer	Boolean	String
start	X	X	X	X
fixed	X	X	X	
min/max	X	X		
nominal	X			

Table 2.1. Some available variable attributes that can be important for the initialization process.

Secondly, Modelica provides the possibility to define variable attributes for each variable. What attributes are actually available depend on their type as listed above.

5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, 19 April, 2013, University of Nottingham, UK.

EOOLT 2013 website:
http://www.eoolt.org/2013/

These attributes affect the initial solution either primarily or secondarily. The usage of the start value depends on the variable type (continuous/discrete) and the fixed attribute.

The start value is used as an initial guess, if `fixed=false`. Otherwise, the start value implicitly generates an initial equation `v=start(v)` for a continuous variable and `pre(v)=start(v)` for a discrete variable.

v(start=v^{start})	fixed=true	fixed=false
type of v: continuous	initial equation: $v = v^{start}$	initial guess of v
type of v: discrete	initial equation: $pre(v) = v^{start}$	initial guess of $pre(v)$

Table 2.2. Interpretation of start attribute depending on fixed attribute and variable kind.

For this reason, it is important to know about the variable type, if the initial condition should be described using these attributes. In Modelica it is not necessary to explicitly declare a variable as discrete or not. This will be automatically detected by a Modelica tool. Due to that reason, it is possible that a variable type is changed in a higher hierarchical component. This can directly affect the corresponding initial equation as introduced above and has to be taken into account during the modeling process.

Using the min and max attribute may restrict the solution space. This can be utilized, for example, to remove physical impractical solutions from the solution space.

The nominal value can be used to setup scaling coefficients. This is the only attribute with no default value. If a Modelica tool detects that there is no nominal value, it can perform some analysis to determine some suitable nominal values by itself.

Finally, Modelica provides the homotopy operator [1] that gives the possibility to formulate actual and simplified expressions for equations. This concept is utilized to improve the convergence properties of the nonlinear iterative solver. A Modelica tool is supposed to introduce the expression (2.1) with a homotopy parameter λ going from 0 to 1.

$$actual \cdot \lambda + simplified \cdot (1 - \lambda) \tag{2.1}$$

3. Mathematical Representation

The mathematical representation will be kept as simple as possible in order to focus on the important aspects. Therefore, some vectors (e.g. inputs) with no effect to the described issues are ignored. With this limitation a hybrid Modelica model can be represented using the following notation described in Table 3.1.

symbol	description
$\underline{x}(t)$	vector of all states
$\underline{\dot{x}}(t)$	vector of all differentiated states
$\underline{y}(t)$	vector of all continuous algebraic variables
$\underline{d}(t)$	vector of all discrete variables
$\underline{p}$	vector of all parameters

Table 3.1. Used symbols for mathematical representation.

The variables $\dot{x}$ and y are unknowns during simulation and are combined to $z(t)$.

$$\underline{z}(t) := \left(\underline{\dot{x}}(t) \quad \underline{y}(t) \quad \underline{d}(t)\right)^{\mathsf{T}} \tag{3.1}$$

The simulation equations can be written as given below.

$$\begin{aligned} f_1\left(\underline{x}(t), \underline{\dot{x}}(t), \underline{y}(t), \underline{d}(t), \underline{d}^{pre}(t), \underline{p}, t\right) &= 0 \\ \vdots \\ f_n\left(\underline{x}(t), \underline{\dot{x}}(t), \underline{y}(t), \underline{d}(t), \underline{d}^{pre}(t), \underline{p}, t\right) &= 0 \end{aligned} \tag{3.2}$$

To efficiently evaluate $\underline{z}(t)$, equations (3.2) are transformed to explicit state space representation (3.3).

$$\underline{z}(t) = \underline{g}\left(\underline{x}(t), \underline{d}^{pre}(t), \underline{p}, t\right) \tag{3.3}$$

The representation (3.3) is not always achievable in an analytic form. But, due to the implicit function theorem such a representation exists, if the corresponding Jacobian is regular. A Modelica tool typically performs the following transformation steps, in order to increase the efficiency. This mathematical representation and the transformation steps of a Modelica model are illustrated using the following example.

```
      model MathRep
x     | Real x1(start=2.0, fixed=true),
      |      x2(start=4);
y     | Real y1, y2, y3(start=-1.5);
d     | Real d1;
      initial equation
h1    | pre(d1) = -0.5 + y1;
      equation
f1    | 0 = -y2 + sin(y3);
f2    | der(x1) = sqrt(x1) + time - d1;
f3    | 0 = x1 + y2 + y3 + 1;
f4    | 0 = x1 + y1 + x1*y1;
      | when {initial(), sample(0.1, 0.1)}
f5    | then
      |   d1 = pre(d1) - y1 + y2;
      | end when;
f6    | der(x2) = x1 + y1;
      end MathRep;
```

Listing 3.1. Example model "MathRep".

Based on a bipartite graph representation of the equation system (see Figure 3.1) a matching algorithm assigns each variable exactly one equation [4].

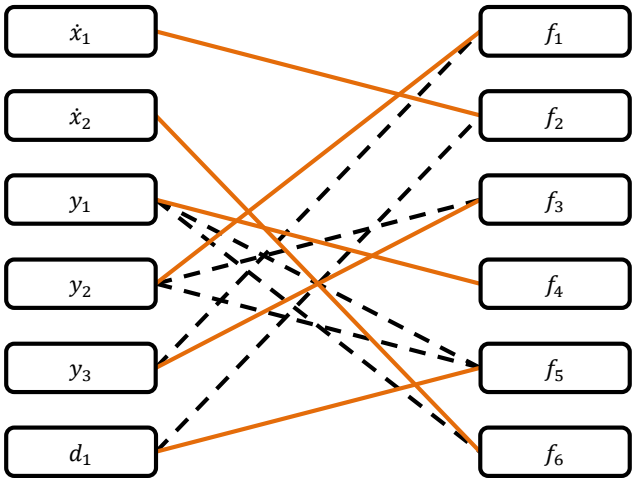

Figure 3.1. Bipartite graph representation and result of the matching for the time-dependent system of example model "MathRep".

The next step is to determine a recursive evaluation order. Due to algebraic loops the result is a block-lower-triangular form (see (3.4)). This is done by determine the strong components using methods like Tarjan's algorithm [5].

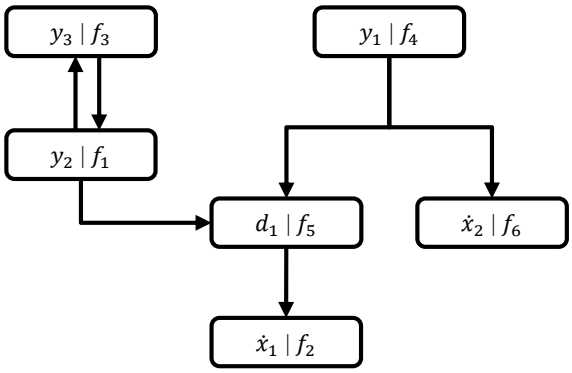

Figure 3.2. Directed graph representation and result of the sorting for the example model "MathRep".

$$\begin{array}{c} \\ f_3 \\ f_1 \\ f_4 \\ f_5 \\ f_6 \\ f_2 \end{array} \begin{array}{c} \begin{array}{cccccc} y_3 & y_2 & y_1 & d_1 & \dot{x}_2 & \dot{x}_1 \end{array} \\ \begin{pmatrix} \boxed{1} & \boxed{1} & 0 & 0 & 0 & 0 \\ \boxed{1} & \boxed{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{1} & 0 & 0 & 0 \\ 0 & 1 & 1 & \boxed{1} & 0 & 0 \\ 0 & 0 & 1 & 0 & \boxed{1} & 0 \\ 0 & 0 & 0 & 1 & 0 & \boxed{1} \end{pmatrix} \end{array} \tag{3.4}$$

More efficiency is gained by so-called tearing algorithms [6], [7], which further reduce the size of algebraic loops.

The principle of the simulation is based on the concept that at a given point in time, especially at the initial time t_0, the states, left limit of discrete variables as well as all free parameters are known. A basic principle is that parameters are constant during simulation and set by the user. Modelica provides via the attribute fixed the possibility that parameters are "free" during the initialization process. The initialization process calculates all needed variables at t_0.

The vector $\underline{\omega}(t_0)$ contains these additional unknowns during initialization. It consists of all states $\underline{x}(t_0)$, all unfixed parameters $\underline{p}^{free}$ and the left limit of all discrete variables $\underline{d}^{pre}(t_0)$.

$$\underline{\omega}(t_0) := \left(\underline{x}(t_0) \quad \underline{p}^{free} \quad \underline{d}^{pre}(t_0)\right)^{\mathrm{T}} \tag{3.5}$$

In order to describe initial conditions additional equations are needed. Mathematically, it is helpful to define the same number of equations than unknowns. If less or more equations than unknowns are given, special treatments are necessary and are described further down.

The initial conditions can be written as illustrated below.

$$\begin{aligned} h_1\left(\underline{x}(t_0), \underline{\dot{x}}(t_0), \underline{y}(t_0), \underline{d}(t_0), \underline{d}^{pre}(t_0), \underline{p}, t_0\right) &= 0 \\ \vdots & \\ h_m\left(\underline{x}(t_0), \underline{\dot{x}}(t_0), \underline{y}(t_0), \underline{d}(t_0), \underline{d}^{pre}(t_0), \underline{p}, t_0\right) &= 0 \end{aligned} \tag{3.6}$$

The goal of the initialization is to determine valid values for $\underline{\omega}(t_0)$. Example models will be discussed within the following sections.

4. Numeric Approach

The numeric approach is the first of two major approaches in OpenModelica to solve the initialization problem. The basic version was already presented in [2] and has been successfully applied in [3].

$$\begin{aligned} &\min_{\underline{\omega}(t_0)} \phi\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right) \rightarrow 0 \\ &\text{s.t.} \\ &\underline{z}(t_0) = \underline{g}\left(\underline{\omega}(t_0), \underline{p}^{fixed}, t_0\right) \\ &\underline{\omega}^{min} \leq \underline{\omega}(t_0) \leq \underline{\omega}^{max} \\ &\text{with} \\ &\phi(.) = \sum_i h_i\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right)^2 \end{aligned} \tag{4.1}$$

The basic idea is to transform the initialization problem into an optimization problem with an optimum that is equal to the initial solution. This is done by interpreting all initial equations as residual ones, which are squared and accumulated to the objective function ϕ. It becomes zero if all equations are satisfied.

4.1 Under/Over-Determined Systems

Often, the modeler misses to fully describe initial conditions to a Modelica model. For the initialization process of such a model it is crucial to provide a determined system. The numeric approach adds additional initial equations by numeric model analysis. Therefore, the Jacobian $\frac{\partial \underline{h}}{\partial \underline{\omega}}\left(\underline{\omega}^{start}\right)$ is numerically approximated and the maximum of the absolute values is selected for each column. Until the initial system is determined the fixed attribute of the variable related to the smallest values are set to true. The heuristics is based on the fact that these variables have the least influence on the initial system near to the start values $\underline{\omega}^{start}$. Therefore, additional initial equations are introduced, which hopefully provide a solvable initial system.

In some cases, Modelica models can be over-determined, e.g. when initial equations are formulated locally in sub-components. If the over-all initial system consists of redundant equations, however fully determines the solution, the current numeric approach can deal with such systems by design [2], [3].

4.2 Scaling

Many problems are hard to solve, since the values of the involved variables are of different magnitudes. To handle such systems variables and equations need to be scaled. In general the nominal attribute $\underline{\omega}^{nom}$ is used for scaling and should be provided by the modeler. By this, the scaling of the variable is straightforward like shown in (4.2).

$$\omega_i^{scaled} := (\omega_i^{nom})^{-1} \cdot \omega_i \tag{4.2}$$

Since the nominal attribute is not available for equations suitable scaling coefficients have to be calculated using differential error analysis. Therefore, each initial equation is approximated by first order taylor expansion. The corresponding scaling factor for each equation is constructed as illustrated in (4.3).

$$\begin{aligned} h(\underline{\omega}) \approx h(\underline{\omega}^{nom}) &+ \underbrace{\frac{\partial h(\underline{\omega}^{nom})}{\partial \omega_1} \cdot \omega_1^{nom}}_{\hat{s}_1^{h_j}} \cdot \frac{\omega_1 - \omega_1^{nom}}{\omega_1^{nom}} \\ &+ \cdots + \underbrace{\frac{\partial h(\underline{\omega}^{nom})}{\partial \omega_n} \cdot \omega_n^{nom}}_{\hat{s}_n^{h_j}} \cdot \frac{\omega_n - \omega_n^{nom}}{\omega_n^{nom}} \\ \tilde{s}_i^h := \begin{cases} |\hat{s}_i^h| & if\ \varepsilon < |\hat{s}_i^h| \\ 1 & else \end{cases} &\ for\ i = 1 \dots n \\ s^h := (max\{\tilde{s}_1^h; \dots; \tilde{s}_n^h\})^{-1} & \\ h^{scaled} := (s^h)^{-1} \cdot h & \end{aligned} \tag{4.3}$$

4.3 Start Value Homotopy

Start Value Homotopy is the name of an extension to the basic numeric approach within OpenModelica. This method uses a different objective function ϕ.

$$\begin{aligned} \phi(.) &= (1-\lambda) \cdot \phi_0 + \lambda \cdot \phi_1 \\ \lambda &\in [0;1] \subset R \end{aligned} \tag{4.4}$$

with

$$\begin{aligned} \phi_0(.) &= \sum_{\forall v} (v - v^{start})^2 \\ \phi_1(.) &= \sum_i h_i\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right)^2 \end{aligned} \tag{4.5}$$

The new objective function is a combination of two sub-objective functions ϕ_0 and ϕ_1. Both are weighted with the homotopy parameter λ (see (4.4)). In the beginning λ is equal to zero and gets increased during the initialization phase until it is one. As a result the initialization algorithm considers in the beginning just ϕ_0 and in the end ϕ_1.

Equation (4.5) shows these sub-objective functions. ϕ_1 is the same function as the objective function of the basic approach. This is quite reasonable, since it should solve the same problem. ϕ_0 is a much simpler function which is based on all explicitly given start attributes.

```
      model forest
x1      Real foxes;
x2      Real rabbits;
y1      Real population(start=350);
y2      Real value;
p       […] // used parameters
      initial equation
h1      der(foxes) = 20;
h2      value      = 11000;
      equation
f1      der(rabbits) = rabbits*g_r -
          rabbits*foxes*d_rf;
f2      der(foxes)   = -foxes*d_f +
          rabbits*foxes*d_rf*g_fr;
f3      population   = foxes+rabbits;
f4      value        = priceFox*foxes +
          priceRabbit*rabbits;
      end forest;
```

Listing 4.1. Example model "forest".

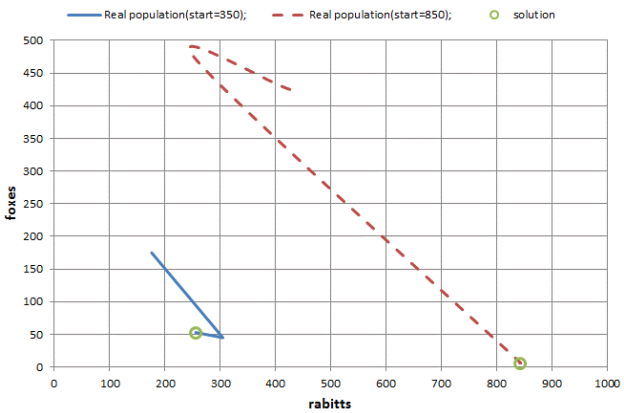

Figure 4.1. Paths within the state space for both initial solutions of the example model "forest".

Figure 4.1 shows the iteration paths for two versions of the example model "forest" from Listing 4.1. The difference between both versions is the start value of `population`. The solid line and the dashed line represent the iteration paths for `population(start=350)` and `population(start=850)`, respectively. The two small circles are exact solutions of the related nonlinear equation system.

The population equation f_3 equally involves both states. In the beginning of the initialization process λ is set to zero, which yields that the objective function consists just of ϕ_0 and considers therefore only the start value of the population. As a result both states are set to half of that start value in the first iteration. As expected both paths continue straight to the corresponding solution.

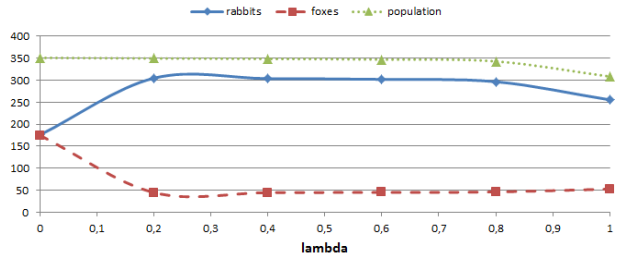

Figure 4.2. Homotopy path with population(start=350).

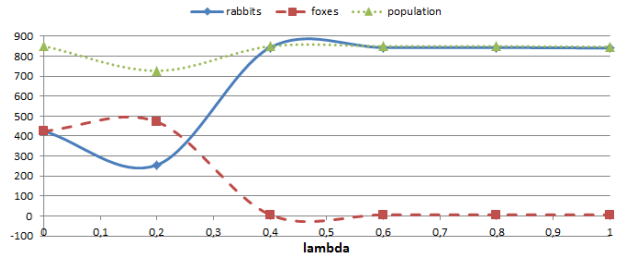

Figure 4.3. Homotopy path with population(start=850).

This numeric approach including the Start Value Homotopy feature has been used as the default initialization method since the end of 2011.

5. Symbolic Approach

This chapter describes the newly developed symbolic approach for solving the initialization problem, which has been investigated and implemented since the beginning of 2012. The main idea behind this approach is the use of symbolic transformation algorithms (matching, sorting, tearing, etc.) that have been sketched in chapter 3 and are already available in the OpenModelica environment. Using the dependence graph with respect to the initialization problem the corresponding equation system is transformed to a block-lower-triangular form. Involved algebraic loops are further reduced by using tearing techniques.

So far, this approach can only be used for determined and under-determined initialization problems. In case of an under-determined initialization problem additional equations are added automatically, based on symbolic model analysis (see section 5.2), until the number of unknowns and equations match. The resulting initialization equation system can finally be described by (5.1) and needs to be solved for the unknowns given by (5.2) during initialization.

$$\begin{aligned} 0 &= f_1\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right) \\ &\vdots \\ 0 &= f_n\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right) \\ 0 &= h_1\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right) \\ &\vdots \\ 0 &= h_m\left(\underline{\omega}(t_0), \underline{z}(t_0), \underline{p}^{fixed}, t_0\right) \end{aligned} \tag{5.1}$$

$$\begin{aligned} \underline{z}(t_0) &:= \left(\dot{\underline{x}}(t_0) \quad \underline{y}(t_0) \quad \underline{d}(t_0)\right)^{\mathsf{T}} \\ \underline{\omega}(t_0) &:= \left(\underline{x}(t_0) \quad \underline{p}^{free} \quad \underline{d}^{pre}(t_0)\right)^{\mathsf{T}} \end{aligned} \tag{5.2}$$

5.1 Dependence Graph

The solution process is firstly described on the example model "forest" that only involves fixed parameters and continuous variables. The result of the matching algorithm performed on the corresponding bipartite graph is presented in Figure 5.1.

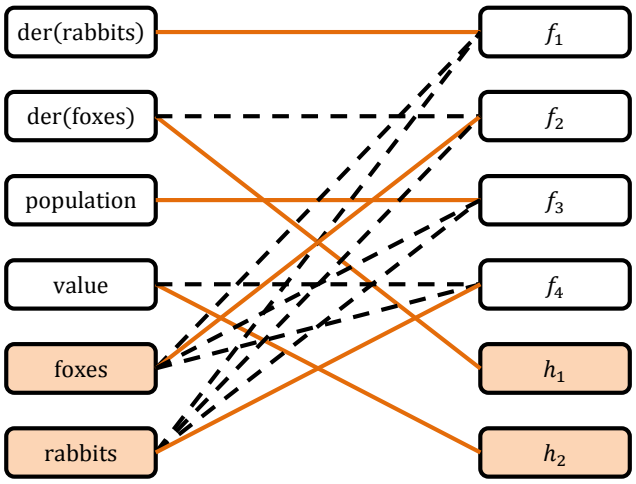

Figure 5.1. Bipartite graph representation and result of the matching for the example model "forest".

Tarjan's algorithm produces the dependence graph below.

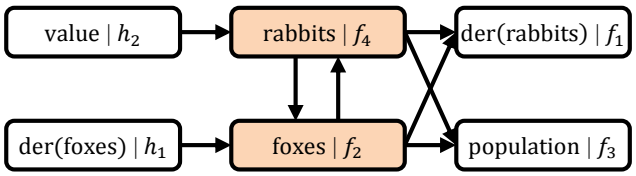

Figure 5.2. Dependence graph of the initialization problem for the example model "forest".

Processing the sorted equation system means that at first the variables $value$ and $der(foxes)$ are calculated in equation h_1 and h_2, respectively. Then, the variables $rabbits$ and $foxes$ are calculated simultaneously through the equation system f_2 and f_4. Finally, the variables $der(rabbits)$ and $population$ are determined using f_1 and f_3, respectively. From this evaluation order it is immediately clear that a starting value for the population will have no influence on the initialization process. This behavior was different when using the Start Value Homotopy approach. In addition, this initialization process is much faster than using the numeric approach. But, the modeler has less influence on the final result of the initialization.

5.2 Under-Determined Systems

As described in section 4.1 it is important to provide a determined equation system for initialization. Since it can happen that the initial conditions are not fully specified, additional equations have to be added to the initialization problem. The symbolic initialization approach in OpenModelica automatically augments these equations based on symbolic model analysis. Additional equations are determined by setting the fixed attribute to true of such components of $\underline{\omega}$ that so far cannot be determined from the initial equation system.

This information can be extracted by processing the sparsity pattern for the Jacobian $\frac{\partial \underline{h}}{\partial \underline{\omega}}$, which can be seen as the collapsed dependence graph of $\underline{\omega}$. If any component of $\underline{\omega}$ cannot be calculated from the initial equation system the whole column is zero. This symbolic method does not depend on $\underline{\omega}^{start}$ as well as other numerical issues compared to the numeric approach.

5.3 Scaling

Using the symbolic approach the initialization problem is transformed to a block-lower-triangular form. As motivated earlier scaling is necessary for finding accurate solutions even when values of variables are of different magnitudes. Same principles are used as described in section 4.2 but only applied on algebraic loops. This reduces enormously the number of Jacobian elements to be calculated.

5.4 Hybrid Models

As mentioned in chapter 3, it is necessary to initialize the continuous as well as the discrete part of a Modelica model. Using the numerical approach the complete hybrid equation system necessary for simulation is considered as constraint for the optimization process. This often leads to a high-dimensional nonlinear optimization problem involving real and discrete variables. Such optimization problems are numerically hard to solve. This issue can be avoided by symbolic transformation steps, which are also used for the simulation.

In the following, the example model "MathRep" from Listing 3.1 will be further analyzed with respect to the initialization apporach. This model contains two states and one discrete variable. Therefore, $\underline{\omega}$ becomes the following:

$$\underline{\omega}(t_0) := \begin{pmatrix} x_1(t_0) & x_2(t_0) & d_1^{pre}(t_0) \end{pmatrix}^{\mathsf{T}} \tag{5.3}$$

Because there are three variables that need to be initialized, it would be necessary that there are also three initial conditions given. The model contains just the initial conditions h_1 as explicitly declared and h_2 (see (5.4)) as implicitly declared. Therefore, the corresponding dependencies from the three unknowns are analyzed and the additional equation h_3 is automatically derived.

h_2	$x_1 = x_1^{start}$	(implicitly declared)	(5.4)
h_3	$x_2 = x_2^{start}$	(automatically declared)	

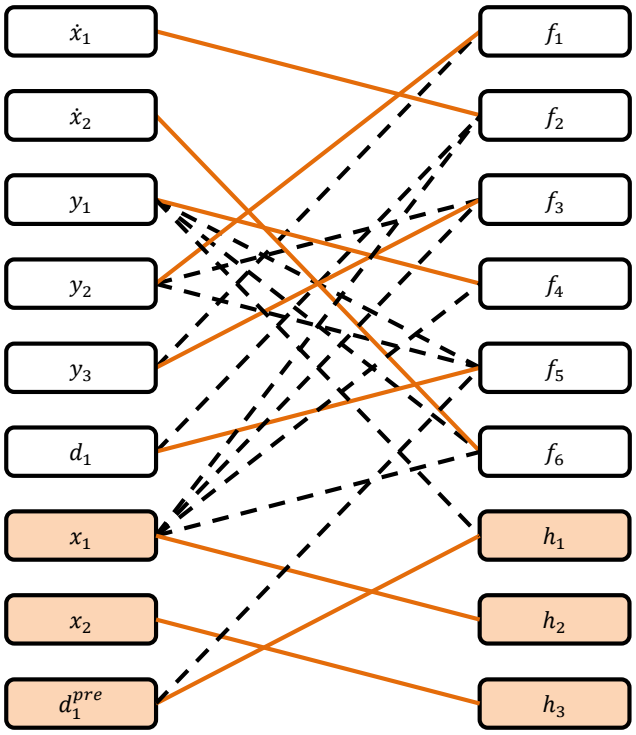

Figure 5.3. Bipartite graph representation and result of the matching for the initial system of example model "MathRep".

Adding this additional information (5.4) to the initial equation system the bipartite graph from Figure 5.3 is generated. Utilizing Tarjan's algorithm the dependence graph presented in Figure 5.4 is produced.

Due to this symbolic approach, the original high-dimensional nonlinear optimization problem involving real and discrete variables is to a large extent reduced to block-lower triangular form.

If corresponding algebraic loops still include real and discrete variables further techniques need to be applied in order to solve these equations. In some cases OpenModelica's tearing heuristic [7] eliminates involved

discrete variables. Same applies, if the involved variables are of boolean or integer type.

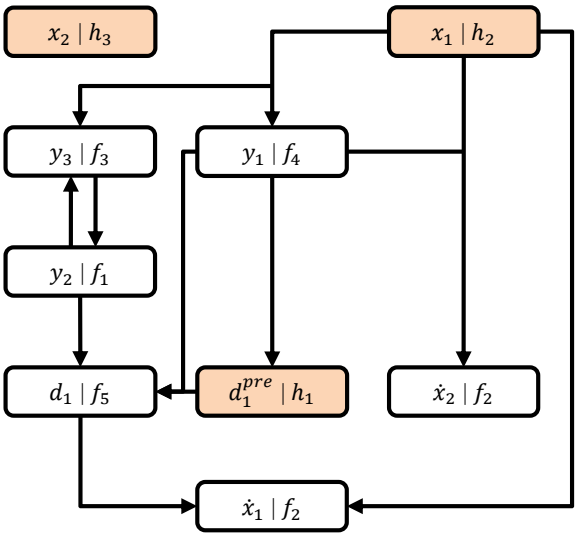

Figure 5.4. Directed graph representation and result of the sorting for the initial system of example model "MathRep".

6. Conclusions and Future Work

This paper describes the principles implemented in the OpenModelica environment, which are utilized to initialize complex hybrid Modelica models. Two major methods, the numeric and symbolic approach, are discussed in detail and advantages and disadvantages have been pointed out.

The numeric approach can deal with over-determined systems and has been successfully applied in [3]. Furthermore, this approach has been extended by the Start Value Homotopy method, which gives the modeler more control on the initialization process.

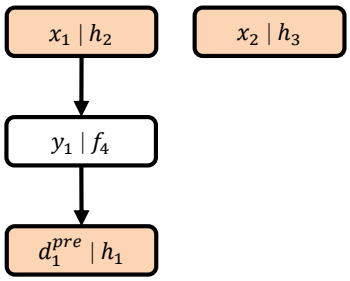

Figure 6.1. Reduced directed graph representation of the initialization problem for the example model "MathRep".

The symbolic approach outperforms the numeric treatment of the initialization problem with respect to performance and solvability in case of large and hybrid systems. With the numeric approach it was so far not possible to initialize the bigger part of model examples in the Modelica Standard Library (MSL). Today, most of MSL examples are initialized efficiently using the symbolic approach.

In case of under-determined initialization problems both approaches introduce additional equations, based on model analysis, in order to generate determined initial systems.

In the future, the two approaches will be more enhanced within the OpenModelica environment. The dependence graph achieved by the symbolic approach can be reduced to represent only the information necessary for determining the initial unknown vector $\underline{\omega}$ (see Figure 6.1 in comparison to Figure 5.4).

Up to now, the Start Value Homotopy method considers all explicitly given start values, which might be not desirable within an object-oriented Modelica drag-and-drop environment. This should be improved by introducing a special Start Value Homotopy annotation keyword. In addition, the Start Value Homotopy feature as well as methods for over-determined systems will be further investigated in order to be integrated into the symbolic approach.

References

[1] Modelica Association, *Modelica® - A Unified Object-Oriented Language for Systems Modeling - Language Specification - Version 3.3*, 2012

[2] Bernhard Bachmann, et.al., *Robust Initialization of Differential Algebraic Equations.* Modelica'2006 Proceedings - Volume 2, pp. 607, 2006.

[3] Francesco Casella, et.al., *Overdetermined Steady-State Initialization Problems in Object-Oriented Fluid System Models.* Modelica'2008 Proceedings - Volume 1, pp. 311, 2008.

[4] Jens Frenkel, et.al., *Survey of appropriate matching algorithms for large scale systems of differential algebraic equations.* Modelica'2012 Proceedings, 2012.

[5] Robert Tarjan, *Depth-first search and linear graph algorithms.* SIAM Journal on Computing, Vol. 1, No. 2, 1972.

[6] Hilding Elmqvist and Martin Otter, *Methods for Tearing Systems of Equations in Object-Oriented Modeling.* Proceedings of the Conference on Modeling and Simulation, eds. Guasch and Huber, pp. 326-332., 1994.

[7] Emanuele Carpanzano, *Order reduction of General Nonlinear DAE Systems by Automatic Tearing*, Mathematical and Computer Modeling of Dynamical Systems. Vol. 6 No. 2, pp. 145-168, 2000.