But where do we start? Qualitative initialization problem with quantitative models

Matthew Klenk and Johan de Kleer and Ion Matei and Daniel Bobrow
Palo Alto Research Center
3333 Coyote Hill Rd
Palo Alto, CA, 94304
klenk,dekleer,matei,bobrow@parc.com

Abstract

For impact in industrial settings, qualitative reasoning techniques must work with existing modeling languages and libraries. This paper presents an important subproblem that must be solved to perform qualitative simulation with industry models, the problem of initialization of systems of hybrid discrete and algebraic equations. We present a case study using five models illustrating the scope of the problem, and then we present three examples of inconsistencies that may arise when guiding the initialization approach with the results of quantitative simulation. We discuss our approach to alleviate these problems in an effort to initialize a very large system model. We close with a discussion of related and future work.

Introduction

As part of DARPA's Adaptive Vehicle Make effort, we seek to deliver qualitative reasoning (QR) tools into the hands of designers. To accomplish this, we face two gaps: (1) the majority of existing QR tools do not operate on models written in the languages used by designers, (2) the majority of QR tools analyze models that are several orders-of-magnitude smaller than those of practicing engineers. Therefore, we are focused on qualitative simulation with models built by engineers for numeric simulation. In previous work, we identified Modelica (Fritzson 2004) as a quantitative modeling language suitable for qualitative reasoning, and we developed the qualitative reasoning module (QRM) as the first automated tool to perform QR analyses of general Modelica models (Klenk et al. 2014b).

Qualitative simulation is central to QRM's analyses of Cyber Physical Systems (e.g., qualitative verification, comparative analysis). To perform qualitative simulation of a Modelica model, it is necessary to translate the hybrid discrete and algebraic equations (DAE) into qualitative constraints, create an initial state, and envision possible outcomes. This paper focuses on the challenge of finding a consistent initial qualitative state. This seems like an extremely easy problem: just set the initial values of variables to the initial values given by the task, e.g., for a pendulum its initial position. Unfortunately, this is not the case — most systems have an enormous number of qualitative initial states. Worse, hybrid DAE's contain conditional equations which are specified to be active only if a particular condition is satisfied. Therefore both qualitative and quantitative hybrid DAE solvers must search across both values and equations to determine a consistent combined set. This paper describes the sources of this ambiguity, the breadth of the problem and our approach to its solution. Ambiguity in the initial situation originates from the following sources:

- It is well known that translating quantitative equations directly into qualitative equations introduces ambiguity (de Kleer and Brown 1984; Struss 1988). Although this source of ambiguity is well understood during envisioning, in our case it leads to additional ambiguity in determining the initial qualitative values, and that in turn cascades into ambiguity in which conditional equations apply initially.
- Most models of real physical systems are described using hybrid DAE's. Hybrid DAE's may have multiple consistent initial states. Determining the best initial *quantitative* values for such systems remains an open research question (Sielemann et al. 2011; Casella 2012). Part of the reason for this is that state variables are not well defined for DAE's (as opposed to linear ODE's).
- Many DAE systems are actually underdetermined. All solvers introduce defaults for undetermined values.

We first illustrate the consequence of the source of ambiguity most familiar to the QR community: the incompleteness of the familiar qualitative algebra (de Kleer and Brown 1984). Later in the paper, we develop the other two sources of ambiguity in more detail. It is well known that the number of variables that need assigned values for all of the variables of a fully determined quantitative model is less than the number of variables that are required to fully determine a qualitative initial state. Consider the following example of a inductor-capacitor oscillator attached to a battery:

$$\frac{di_L}{dt} = \frac{v_L}{L} \tag{1}$$

$$\frac{dv_C}{dt} = \frac{i_L}{C} \tag{2}$$

$$v_L = v_B - v_C \tag{3}$$

A quantitative model could determine the values of all the variables with the following set of initial values ($L =$

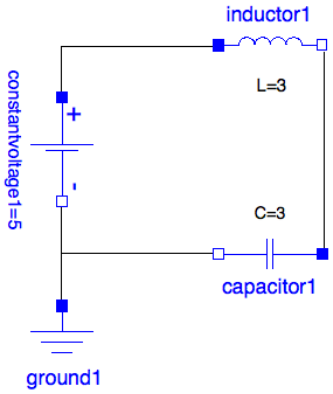

Figure 1: Modelica model of an inductor-capacitor system

$3, C = 3, v_B = 5, i_L = 2, v_C = 5$). Where L and C refer to the inductance and capacitance, v and i refer to voltage and current, and the subscripts refer to the component over which the measurement is made with B, L, and C refer to the battery, inductor and capacitor. Now, taking the sign algebra qualitative abstractions for each of these initial values and then propagating their values through the qualitative constraints equivalent to above equations results in unknown qualitative values for $(v_L, \frac{di_L}{dt})$. This is due to ambiguity in qualitative arithmetic for equations 1 and 3. Searching for all possible values (3^2) combinations) results in three initial consistent qualitative states.

This is a significant problem for two reasons: (1) the search for possible initial values is exponential in the number of free variables, and designers using QRM work with simulation models with 100s to 10,000s of equations. (2) The ambiguity may occur in conditional expressions in the quantitative model, in which case the search for an initial qualitative state must also examine the space of possible qualitative constraints.

After providing the relevant background on the Modelica modeling language, qualitative simulation, and our QRM System, we explore the scope of this initialization problem using a set of models. Next, we describe an approach to this problem that specializes the qualitative model by using the results of quantitative simulation to determine a consistent initial state. Unfortunately, this results in qualitative inconsistencies that we demonstrate through three examples. We close with a discussion regarding the proper treatment of quantitative simulation results for use in qualitative models.

Background

DAEs in the Modelica Language

Figure 2 illustrates the modeling and simulation process when using Modelica tools. The process begins with the user creating a hierarchical Modelica model from a library of reusable components. Next, the tool compiles the composed model into a set of possibly hybrid differential algebraic equations (DAE) and uses numerical integration methods, e.g., DASSL (Petzold 1982), to produce a sequence of

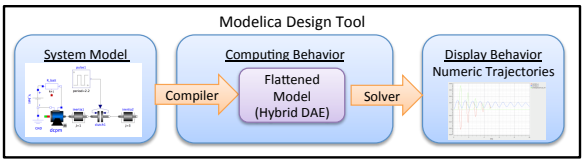

Figure 2: Modelica tools allow designers to compose models from existing libraries of components and define new components. The composed system can then be simulated numerically over time.

numeric values for every model variable over time. In Modelica, the continuous-time behavior is governed by differential (4) and algebraic (5) equations with 'state' variables, x, algebraic variables, z, inputs, u, and output variables, y. The algebraic variables may include discrete variables, such as booleans or enumerated types.

$$\dot{x} = f(x, u, z) \tag{4}$$
$$g(x, u, z) = 0 \tag{5}$$
$$y = h(x, u, z) \tag{6}$$

Most of qualitative reasoning is developed in the context of differential equations where the equations that govern system behavior can be written as:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

where x are the system state variables, u are in the inputs, y are the output variables and A, B, C and D are matrices whose entries are functions of system parameters.

In DAE's, the equations can include algebraic constraints (i.e., Equation 4 includes z on the right hand side). A simple example is the planar pendulum (Figure 3):

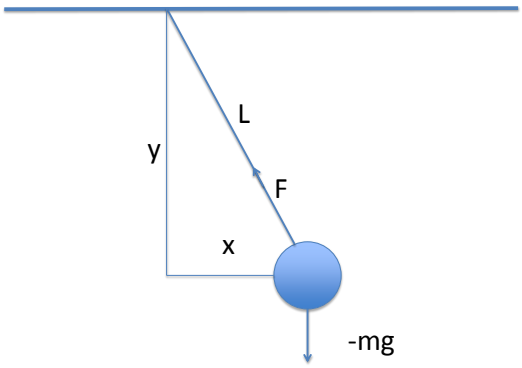

Figure 3: Planar pendulum.

$$m\dot{v_x} = -\frac{x}{L}F$$
$$m\dot{v_y} = -\frac{y}{L}F - mg$$
$$\dot{x} = v_x$$

$$\dot{y} = v_y$$
$$x^2 + y^2 = L^2$$

This last equation introduces an algebraic constraint. These equations can be turned into a simpler set by a change of variable, but such change of variables rarely apply to real DAE's.

Discrete Events in Modelica

In addition, Modelica models include a set of conditions that trigger discrete changes, or *events*, in the system. Events are simulated as though they were instantaneous. They may change the values of variables as well as the set of equations governing the behavior of the system. Furthermore, events may cause other events. While sequential, the entire event sequence has no duration.[1] Events may occur during initialization as well as immediately after. Therefore the initial values of the simulation may differ from the values of the system at $t = 0$.

The following demonstrates a Modelica model for a ball bouncing on a flat surface. Notice that when the ball drops below the surface (the test `h<0`) the velocity is reversed. This is a very simple example of a Modelica model in which normally continuous variables can vary discontinuously.

```
model BouncingBall "The 'classic'
    bouncing ball model"
  type Height=Real(unit="m");
  type Velocity=Real(unit="m/s");
  parameter Real e=0.8 "Coefficient of
      restitution";
  parameter Height h0=1.0 "Initial
     height";
  Height h;
  Velocity v;
initial equation
  h = h0;
equation
  v = der(h);
  der(v) = -9.81;
  when h<0 then
    reinit(v, -e*pre(v));
  end when;
end BouncingBall;
```

These discontinuous changes may also result in changes to the DAE itself. In the brake model below, the equation that governs the force applied by the brake `f` is governed by five conditional expressions with five discrete variables `locked, free, startForward, startBackward, mode`.

```
f = if locked
 then sa*unitForce
 else if free
  then 0
  else cgeo*fn*
(if startForward
   then tempInterpol1( v, mue_pos,2)
   else
   if startBackward
     then -tempInterpol1(-v,
        mue_pos, 2)
     else
     if pre(mode) == Forward
       then tempInterpol1( v,
          mue_pos, 2)
       else -tempInterpol1(-v,
          mue_pos, 2));
```

Quantitative Model Initialization

Here we provide an overview of the Modelica model initialization process for more detail see (Mattsson et al. 2002). Given a set of possibly hybrid differential and algebraic equations and a possibly incomplete set of initial values, the solver must determine a consistent set of values for all variables, derivatives, and pre-variables (i.e., the values of discrete variables before simulation begins). In object oriented modeling, the modeler typically works with individual components that are composed to create the system under test. The modeler cannot envision every use of the component model and, therefore, cannot specify all of the initial conditions. Modelica compilers (e.g., OpenModelica[2]) search for a solution to the initial equations using a combination of numeric (Bachmann, Aronsson, and Fritzson 2007) and symbolic methods (Ochel and Bachmann 2013).

Simulation Modeling

It should come as not surprise that, at an abstract level, DAE solvers operate much like envisioners. A DAE solver will identify every changing variable which is used in any condition. It will create a variable for every condition such that the event happens at zero-crossing of that variable. Then the integrator will search forward to find the first zero crossing. Most envisioners operate within the same framework except they often cannot distinguish which of the changing variables changes first and therefore have to consider all possible ordering of changes, instead of finding the numerically earliest zero crossing.

Qualitative Ambiguity in QRM

QRM performs qualitative analyses of Modelica models by abstracting the hybrid-DAE into a set of constraints, finding a set of initial qualitative states, and performing an envisionment that includes the continuous as well as discrete-time behaviors defined by Modelica (Klenk et al. 2014b).

Figure 4 provides an overview of this process. QRM begins by using OpenModelica to flatten the model and produce an XML representation of the hybrid DAE. Next, QRM abstracts this set of equations into conditional constraints. Then, QRM creates a partial state using qualitative abstractions of the quantitative initial conditions. Through constraint satisfaction, QRM creates set of consistent qualita-

[1] It is worth noting that this approach is very similar to early QR approaches that employed infinitesimals and inconsistent *mythical instants* to account for discrete changes (Nishida and Doshita 1987)

[2] www.openmodelica.org

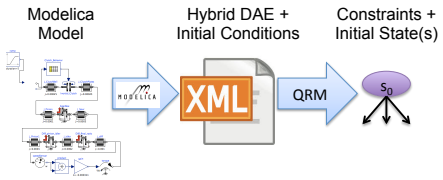

Figure 4: Modelica to Qualitative Initialization

tive states. From these states, QRM generates an attainable envisionment.

Qualitative Simulation Semantics

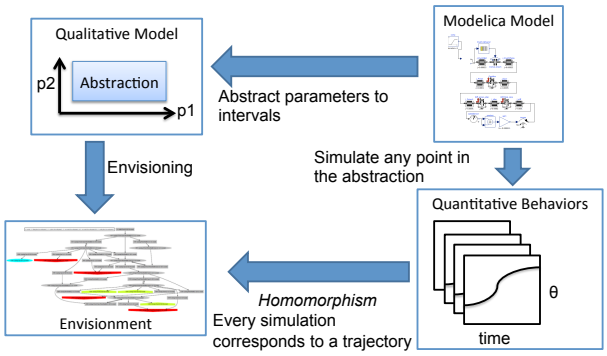

Figure 5: Desired semantics for qualitative simulations with Modelica models

The desired semantics of a qualitative simulation of a Modelica model are that the results of every quantitative simulation of consistent sets of numeric parameters within the parameter intervals of the qualitative model correspond to a trajectory in the envisionment. This relationships is shown in Figure 5. Given a Modelica model (upper right), we create an abstraction consisting of constraints and an initial qualitative state (upper left) from which we produce an envisionment (lower left). Every assignment of parameters that is consistent with respect to the Modelica model and the abstraction will result in a quantitative simulation (lower right), and each correct simulation will correspond to a trajectory in the envisionment.

Thus, for initialization, the initial qualitative state(s) should align with the results of quantitative simulation at $t = 0$.

Qualitative Model Initialization

As shown in Figure 4, QRM uses the XML representation of the hybrid DAE produced by OpenModelica to construct an initial set of qualitative states. This representation includes equations that are abstracted into qualitative constraints in the same manner as QSIM (Kuipers 1994) with one significant addition. Modelica equations may include conditional expressions, which require conditional constraints as described in (Klenk et al. 2014b). For the purposes of this paper, the important aspect of initialization is that the XML representation also includes the modeler defined quantitative initial conditions. QRM defines the qualitative value for the corresponding variable(s) and then propagates these values through the constraint network. For the remaining unknown variables, QRM searches for all possible values that are consistent with the constraints.

Case Study

To demonstrate the scope of this problem, we present data collected from a series of Modelica system models. These models were used in the development and evaluation of QRM. For this study, we include toy problems developed internally and used for explanatory purposes and testing as well as models developed by Modelon for the Adaptive Vehicle Make DARPA program.

LC_Battery Inductor capacitor model shown in Figure 1

HVAC_Dry Model that converts humidity into total dry air mass

Driveline Driveline model that includes an engine, clutch, gear ratio, air resistance, and the road

Waterjet Model of the propulsion system of an amphibious vehicle

FullDriveline Detailed driveline model containing multi-physics for the DARPA Adaptive Vehicle Make program

For each model, we report the following metrics:

- Number of Modelica equations is the number of equations in the flattened model after all OpenModelica simplifications
- Number of underspecified quantitative variables (as reported by the OpenModelica compiler) is the number of variables that the compiler must assume values to arrive at an initial state
- Number of qualitative variables is the number of qualitative variables after QRM imports and simplifies the model
- Number of unknown qualitative variables after initialization with quantitative initial conditions

Model	Modelica Equations	Under-specified Variables	Qualitative Variables	Qualitative Unknown Variables
LC_Battery	3	0	11	2
HVAC_Dry*	4	0	27	5
Driveline	50	4	102	35
Waterjet*	76	22	365	257
FullDriveline *	1685	55	3512	782

Table 1: Initial quantitative and qualitative ambiguity. * indicates models developed by Modelon AB.

The results in Table 1 illustrate that this problem occurs in large and small models as well as models authored by engineers familiar with Qualitative Reasoning. In the worst case, the number of possible initial qualitative states is exponential in the number of unknown qualitative variables. For example, the Waterjet propulsion system consists of 76 Modelica equations has a space of 3^{257} possible initial qualitative states. In the following sections, we discuss how the

results from quantitative simulation can be used to reduce this ambiguity along with additional issues that arise from this combination.

Specializing the Qualitative Model with the results of Quantitative Simulation

The ambiguity in the initial qualitative state arises from qualitative arithmetic as well as the quantitatively underdetermined system of equations with unknown states variables, state derivatives, and algebraic variables. Because the Modelica compiler does not suffer from the first source and must overcome the second source of ambiguity, it is appealing to determine the initial qualitative state by abstracting the results of the quantitative simulation at $t = 0$. Figure 6 illustrates the steps of this process.

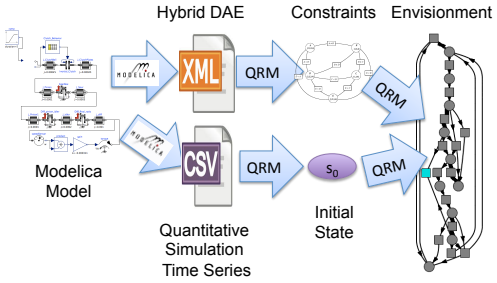

Figure 6: Our initialization process uses the OpenModelica in two ways. The OpenModelica compiler produces an XML representation of the hybrid DAE as well as time series for each of the model variables stored in a CSV file.

Unfortunately, OpenModelica will not report values exactly at $t = 0$. Modelica allows events to occur with no duration. Hence, the first set of values we receive from Modelica are at $t = 0 + \delta$ even though the CSV file reports it as $t = 0$. One consequence of this is that QRM cannot rely on *any* initial value provided by the designer because the solver may have already discontinuously changed any variable within the set of events occurring at $t = 0$. When using the results of quantitative initialization, QRM only relies on the initial values provided by the solver at $t = 0 + \delta$ and ignores any initial values provided by the designer.

Another larger challenge is that quantitative initialization frequently includes errors due to floating point precision. When using these values to determine the initial set of constraints, it is possible to arrive at an inconsistent initial situation. In the rest of this section, we illustrate this behavior through a series of examples.

Example: Power Takeoff Module

Consider the Modelica equation in Figure 7 from a model of a cross drive with power takeoff module.

This equation is used to calculate the temperature difference that will be used to compute the heat flow through the component. `noEvent()` is a Modelica operator that tells the simulation engine to not attempt to compute the exact time at which the argument crosses zero. OpenModelica initialization computes the values shown in Table 2.

```
if noEvent((heatport.T-T_a)*(
   heatport.T-T_b)<0.0) then
delta_t = heatport.T - t_{avg};
else
if noEvent(abs(heatport.T-T_a)>abs(
   heatport.T-T_b))
   then delta_t = heatport.T-T_b
   else delta_t = heatport.T-T_a
end if;
end if;
```

Figure 7: Power takeoff module

Variable	Value
`delta_t`	7.86E-11
`heatport.t`	353.15
`T_a`	353.15
`T_b`	353.15
`T_avg`	353.15

Table 2: Initial values from OpenModelica for the Cross Drive Power Takeoff Module

The constraints resulting from abstracting the equation in Figure 7 are inconsistent with the sign algebra abstraction of the values in Table 2. The qualitative variable representing the difference between 353.15 and 353.15 is Q0 and it must be equal to the qualitative value of `delta_T` which is Q+.

Example: Quadratic Fluid Flow

The XML hybrid DAE of System Design Test, a detailed vehicle driveline model, contains the following two equations:

```
dP = if  hyd_a.p - pressure_Source1.p
   >= 0.0
        then hyd_a.p -
           pressure_Source1.p
        else pressure_Source1.p -
           hyd_a.p

outflow = if hyp_a.p >
   pressure_Source1.p
          then pressure_source2.ports
             [1].outfow
          else 0.0
```

Rewriting these using single letters for each Modelica variable name the form because clearer:

```
d = if  h - p >= 0.0
        then h - p
        else p - h

o = if h > p then q else 0.0
```

Notice that both Modelica equations are comparing the same h and p. We can see that the first equation computes the absolute value of $h - p$. The values provided by OMC initializes $h = p = 1d7, q = 41960$. Substituting these into

the Modelica equations we see that the initial values for d and o must be $d = 0, o = 0$. Modelica solvers are based on numeric integration not symbolic algebra, therefore the exact result depends on the details of the numerical integrator used. Unfortunately, the OpenModelica, which we use, obtains $d = 0$ and $o = 41960$.

These values are both numerically and qualitatively inconsistent. This inconsistency is not the result of some round off error, so a new approach must be used to handle this case. It is easy to criticize OMC for this error, but that solver usually produces correct results and this type of inconsistency happens regularly. Its simply the result of numerical integrators updating the pseudo-state variables at slightly different instants. The actual curves produced by OMC are correct, they just has a wrong value at $t = 0$.

Example: Step Function

Modelers frequently modify their models of the physical world to aid the solvers they expect to be analyzing their models. Consider the following step input function:

```
y=if(x>0) then y1 else y2;
```

Figure 8: Step input function

In large models, modelers may choose to approximate this relationship to ensure continuity using the Modelica `smooth` operator.

```
y = smooth(1,
           if x > x_small
              then y1
              else if x < -x_small
                   then y2
                   else f(y1, y2));
```

Figure 9: Use of Modelica `smooth` operator

In the region `-x_small < x < x_small` a 2nd order polynomial, f, is used for a smooth transition from $y1$ to $y2$. Directly translating Equation 9 introduces three additional modes into the model that are only important for the quantitative solver. Meanwhile, the equation in Figure 8 is more appropriate for qualitative reasoning. Therefore, OpenModelica includes rewrite rules that automatically translate equation in Figure 9 into the equation in Figure 8 when writing the XML representation of the DAE. While the qualitative constraints resulting from equation in Figure 8 indicate that y can only have one of two values $y1$ or $y2$, if we use the initial values from the OpenModelica simulation, y could be any value in the interval $[y1, y2]$ resulting in an inconsistency. To make matters worse, the modeler may choose a value for `w_small` that is not small and use the `smooth` operator as an interpolation function, as occurs many times in the Fluid package of the Modelica Standard Library.

Current Approach

In this section, we describe the approach we are currently using and its limitations. Due to the possibility of discrete events occurring during initialization, QRM starts with the solver provided values for all variables at $t = 0^+$ and ignores all initial assignments provided by the designer in the XML representation of the DAE.

input : constraints, C, and real values for variables, R
output: a consistent initial qualitative state, S
repeat
  $S \leftarrow$ relaxedAbstraction(R);
  $S \leftarrow$ runConstraints(C,S);
  foreach $c \in$ *clusters(S)* **do**
    if $\neg$ *solution(c)* **then**
      $R \leftarrow$ retractInitialConditions(S,C,R);
    end
  end
until *solution(S)*;

Algorithm 1: Initialization procedure of qualitative state from quantitative simulation results

Algorithm 1 provides an overview of this process. The first step is to create a relaxed abstraction in which each real valued variable that is not close to a threshold is assigned its respective qualitative value. This results in a partial state, as many variables will have unknown qualitative values. Next, we separate the set of unknown variables into the maximal number of clusters satisfying the following criteria. Within each cluster, the constraints of all of the unknowns reference only known variables or variables within the cluster. Next, we attempt to find a solution to the constraints for each cluster. If we find one, the values for these variables are added to the partial situation, if not, then we resolve the inconsistencies by performing model-based diagnosis to determine the minimum number of initial qualitative values to retract. Once a solution to each cluster has been found, the partial state is now a complete consistent initial qualitative state.

Relaxed Abstraction

Many of the qualitative inconsistencies arise from the abstraction of near-zero quantitative values. Ideally we would like to find an ϵ such that:

$$x < -\epsilon \rightarrow [x] = Q-$$
$$x > \epsilon \rightarrow [x] = Q+$$
$$|x| \leq \epsilon \rightarrow [x] = Q0$$

Unfortunately, we found no value for $\epsilon < 1$ that is sufficient to remove qualitative inconsistencies across our model suite (e.g., the contradiction in quadratic fluid flow example above cannot be resolved in this way). Therefore, we make values below the threshold ambiguous (i.e., $|x| \leq \epsilon \rightarrow [x] = Q?$). Choosing $\epsilon = 10^{-4}$ ensures there are relatively few inconsistencies.

It is important to note that we have to apply the same rule to system parameters (e.g., resistances, forces and masses)

and floating point constants in equations as well. The fundamental reason for this is that Modelica preprocesses equations involving parameters and this too introduces errors. For instance, Modelica may have simplified the equation $10.0 - 10.0$ to 0.0. If these two constants resulted from subtracting the masses of two blocks then we cannot guarantee that their weight difference is exactly 0 ($Q0$). If these two constants resulted from subtracting the mass of a block from itself, then the result should be exactly 0. Since Modelica does not inform QRM the source of any constant in any equation, QRM needs to assume small constants are $Q?$ as well.

Of the 3512 qualitative variables in System Design Test, relaxed abstraction assigns values to all but 782.

Cluster unknown variables

After performing constraint propagation over the initial values from the previous step, there will still be unknown variables. To find a consistent assignment of values for these unknowns, we take a divide and conquer approach. We divide the unknowns into mutually constrained clusters. That is, for each constraint that mentions more than one unknown, all of the mentioned unknowns are in the same cluster.

Consider two equations $x + y + z = 0$ and $z + r + v = 0$. These two constraints share a variable z, so solving them requires searching 3^5 space of variable assignments if all are initialized to $Q?$. However, if z has a known qualitative value, then those two equations are independent. Each is a 2 variable system which can be solved faster $2^3 + 2^3$.

For System Design Test, the 782 unknown variables are divided into 84 disconnected clusters. After removing the inconsistencies, QRM finds 80 clusters with 819 unknown variables. The increase in unknown variables is due to the fact that removing inconsistencies removes constraints from the model.

Retract inconsistent initial conditions

Within a cluster, we iterate between assigning a qualitative value to an unknown variable and propagating the results to find a solution. If there is no consistent qualitative value for a variable, we use model-based diagnosis (de Kleer and Kurien 2003) to determine the minimal initial conditions that must be retracted.

QRM takes the following steps to detect the inconsistency of the Quadratic Fluid Flow model. The following summarizes the result: (1) From the second equation QRM deduces that $[h - p]$ must be Q0 or Q-. (2) From the first equation if $[h - p] = Q0$ then $[d] = Q0$, (3) Again from the first equation if $[h - p] = Q-$ then $[d] = Q0$. Therefore $[d] = Q0$ in all cases.

Once a consistent assignment of qualitative values to all unknowns in the cluster have been found, the cluster is solved. Once all of the clusters have been solved, then their union with the results of relaxed abstraction provides a consistent initial qualitative state.

Given enough time and space, this approach will always find an initial qualitative state if the initial constraint set is consistent.

OMC v. Dymola

Another complication is that different Modelica compilers will result in different solutions to the initialization problem. Consider again the quadratic fluid flow example. Unlike OpenModelica, Dymola obtains the values we expect: $d = 0, o = 0$. We can only speculate why Dymola and OMC obtain different answers here. Some reasons might be that one of them computes the floating point operations in different order than the other. There is a very large discontinuity in o (it is has only two legal values 41960 and 0 here). Another reason might originate from the fact that $h > p$ and $h - p > 0.0$ are not equivalent in floating operations (the computation of $h - p$ may produce an underflow which would be interpreted as 0 by the software, while a floating point comparison of h and p shows h is greater than p).

Related Work

Aside from a few notable exceptions (e.g., (Struss and Price 2004)), QR has not been applied to models created by engineers and designers in industrial settings. Qualitative deviation models can be abstracted from Modelica differential equation models (Struss et al. 2014). Their work succeeds by placing strong constraints on how models are represented in Modelica. Deviation models are primarily useful for diagnosis, while our work focuses on qualitative analyses based on simulation.

Conclusions

Our experience here illustrates the difficulties in qualitative model initialization when working with real world models built by industrial designers. We were able to initialize and envision Modelica models with hundreds of variables. For example, the driveline model above results in an envisionment with depth 53, $>$ 5 million states, and $>$ 17 million edges.

Our work with Modelica has found number of features that support numeric simulation, but complicate formal methods (Klenk et al. 2014a), and the qualitative abstraction described here requires a more rigorous analysis to prove the relationship between the qualitative and quantitative models.

One theme we see in this work is the need to understand the assumptions of use in which models are constructed. One approach to this involves meta-modeling (Sangiovanni-Vincentelli et al. 2009) which seeks to capture shared concepts at a meta-level allowing different design tools to perform their analyses and providing a unified view of the results.

Acknowledgments

Material within this technical publication is based upon the work supported by the Defense Advanced Research Projects Agency (DARPA) as part of a subcontract under Vanderbilt University Prime Contract HR0011-13-C-0041. The views, opinions, and/or findings, contained in this article are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense

or the U.S. Government.[3]

References

[Bachmann, Aronsson, and Fritzson 2007] Bachmann, B.; Aronsson, P.; and Fritzson, P. 2007. Robust initialization of differential algebraic equations. In *EOOLT*, 151–163.

[Casella 2012] Casella, F. 2012. On the formulation of steady-state initialization problems in object-oriented models of closed thermo-hydraulic systems. In *Proceedings of the 9th International Modelica Conference, Munich, Germany, September 3*, volume 5.

[de Kleer and Brown 1984] de Kleer, J., and Brown, J. S. 1984. A qualitative physics based on confluences. *Artificial Intelligence* 24(1):7–84.

[de Kleer and Kurien 2003] de Kleer, J., and Kurien, J. 2003. Fundamentals of model-based diagnosis. In *Proceedings of the 5th IFAC Symposium SAFEPROCESS*, 25–36.

[Fritzson 2004] Fritzson, P. 2004. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Piscataway, NJ: Wiley-IEEE Press.

[Klenk et al. 2014a] Klenk, M.; Bobrow, D.; de Kleer, J.; and Janssen, B. 2014a. Making modelica applicable for formal methods. In *Proceedings of the 10th International Modelica Conference*.

[Klenk et al. 2014b] Klenk, M.; de Kleer, J.; Bobrow, D.; and Janssen, B. 2014b. Qualitative reasoning with modelica models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[Kuipers 1994] Kuipers, B. 1994. *Qualitative reasoning: modeling and simulation with incomplete knowledge*. Cambridge, MA, USA: MIT Press.

[Mattsson et al. 2002] Mattsson, S. E.; Elmqvist, H.; Otter, M.; and Olsson, H. 2002. Initialization of hybrid differential-algebraic equations in modelica 2.0. In *2nd Inter. Modelica Conference 2002*, 9–15.

[Nishida and Doshita 1987] Nishida, T., and Doshita, S. 1987. Reasoning about discontinuous change. In *Proc. AAAI*, volume 87, 643–648.

[Ochel and Bachmann 2013] Ochel, L., and Bachmann, B. 2013. Initialization of equation-based hybrid models within open-modelica.

[Petzold 1982] Petzold, L. R. 1982. Description of dassl: A differential/algebraic system solver. Technical report, Sandia National Labs., Livermore, CA (USA).

[Sangiovanni-Vincentelli et al. 2009] Sangiovanni-Vincentelli, A.; Yang, G.; Shukla, S. K.; Mathaikutty, D. A.; and Sztipanovits, J. 2009. Metamodeling: An emerging representation paradigm for system-level design. *Design & Test of Computers, IEEE* 26(3):54–69.

[Sielemann et al. 2011] Sielemann, M.; Casella, F.; Otter, M.; Clauß, C.; Eborn, J.; Mattsson, S. E.; and Olsson, H. 2011. Robust initialization of differential-algebraic equations using homotopy. In *Proceedings of the 8th Modelica Conference*, 21–22.

[Struss and Price 2004] Struss, P., and Price, C. 2004. Model-based systems in the automotive industry. *AI Magazine* 24(4):17–34.

[Struss et al. 2014] Struss, P.; Sterling, R.; Febres, J.; Sabir, U.; and Keane, M. M. 2014. Combining engineering and qualitative models to fault diagnosis in air handling units.

[Struss 1988] Struss, P. 1988. Mathematical aspects of qualitative reasoning. *International Journal of Artificial Intelligence in Engineering* 3(3).

[3]DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.