Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Qualitative Reasoning with Modelica Models

Matthew Klenk, Johan de Kleer, Daniel G. Bobrow, Bill Janssen
Palo Alto Research Center
3333 Coyote Hill Rd
Palo Alto, CA, 94303
klenk,dekleer,bobrow,janssen@parc.com

Abstract

Qualitative reasoning can play an important role in early stage design. Currently, engineers explore the design space using simulation models built in languages such as Modelica. To make qualitative reasoning useful to them, designs specified in their languages must be translated into a qualitative modeling language for analysis. The contribution of this paper is a sound and effective mapping between Modelica and qualitative reasoning. To achieve a sound mapping, we extend *envisioning*, the process of generating all relevant qualitative behaviors, to support Modelica's declarative events. For an effective mapping, we identify three classes of additional constraints that should be inferred from the Modelica representation thereby exponentially reducing the number of unrealizable trajectories. We support this contribution with examples and a case study.

1 Qualitative reasoning and design

Qualitative reasoning (QR) (Kuipers 1994)(de Kleer and Brown 1984)(Forbus 1984), which automates reasoning about the continuous world using abstraction, can play an important role in early stage design. Unfortunately, the languages of QR have not made inroads into engineering practice, and, consequently, QR has not been applied in industrial settings, with a few notable exceptions (e.g., (Struss and Price 2004)). Instead, engineers use simulation models built in languages such as Modelica (Fritzson 2004), to understand the possible behaviors of their design. To make QR useful to them, it is necessary to operate directly from the engineer's models.

QR captures the relevant differences in behavior resulting from significant differences in parameter values. Thus, QR can play an important role in early stage design. We work to realize this promise as part of the DARPA Adaptive Vehicle Make[1] (AVM) program. AVM seeks to dramatically reduce the cost and time required to design, verify and manufacture complex cyber-physical systems. We integrated qualitative reasoning into the CyPhy toolchain (Simko et al. 2012) for use by designers (Lattmann et al. 2014). The CyPhy toolchain uses the Modelica language to model hybrid systems. When a designer runs a simulation to determine if a model will meet a particular requirement, one of two things will happen. One, the model fails to meet the requirement. In this case, QR can inform the designer which parameters to change or if a change in structure is required. Two, the model meets the requirement. In this case, QR can help the designer understand what ranges of parameter values will maintain success, and under what circumstances lead to failures.

The contribution of this paper is a sound and effective mapping between Modelica and QR. We create an abstraction of the Modelica model. From this abstraction, our *envisioning* algorithm generates the set of all possible qualitative behaviors, called an *envisionment*. In a sound mapping, the behavior of every Modelica model with same structure of components corresponds to a trajectory in the envisionment. Unlike current QR approaches, Modelica events are specified declaratively with acausal component models. This confers two advantages: (1) the models correspond to the physical structure of the system, and (2) by being declarative and acausal, they are easier to reuse in different contexts.

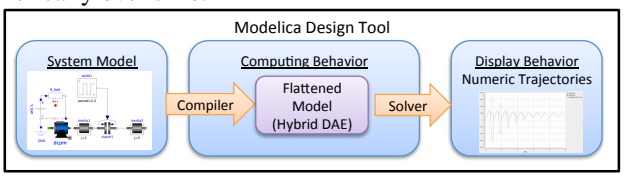
To create a sound mapping, we introduce the first envisioning algorithm for computing the qualitative consequences of Modelica events. Central to this approach is the role of non-discontinuous variables across events. Unfortunately, directly abstracting Modelica models results in a large number of qualitative behaviors that are not realizable, that is, they have no corresponding quantitative behavior. While these *spurious* behaviors (Struss 1988) are well studied in the literature, reducing them improves the utility of the resulting analyses. We identify three classes of additional constraints that must be computed from the Modelica equations. We demonstrate this contribution through detailed examples, a case study that shows the additional constraints result in an exponential reduction in the size of the envisionment.

2 Modelica models and qualitative reasoning

To abstract Modelica models for envisioning, it is necessary to understand how each approach represents continuous and discrete dynamics. This allows us to define a sound qualitative abstraction of Modelica models. While Modelica's continuous behavior corresponds directly with existing envisionments, there is no such alignment between current

[1] http://en.wikipedia.org/wiki/Adaptive_Vehicle_Make

Figure 1: 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.



QR approaches and Modelica's declarative events. To account for this discrete behavior, we introduce an algorithm for computing the qualitative effects of events using *conditional* constraints and show how it corresponds with Modelica.

2.1 Dynamics modeling in Modelica

Figure 1 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 hybrid differential algebraic equations (DAE) and uses numerical methods, e.g., DASSL (Petzold 1982), to produce a sequence of numeric values for every model variable over time. In Modelica, the continuous-time behavior is governed by differential (1) and algebraic (2) equations with state variables, x, algebraic variables, y, and inputs, u. The algebraic variables may include discrete variables, such as booleans or enumerated types.

$$\dot{x}(t) = f(x(t), y(t), u(t)) \quad (1)$$

$$y(t) = g(x(t), u(t)) \quad (2)$$

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.

2.2 Dynamics modeling in qualitative reasoning

Our constraint-based qualitative reasoner draws on the ideas from established methods (de Kleer and Brown 1984)(Kuipers 1994). QR abstracts continuous quantities into intervals bounded by *landmarks*. For example, the qualitative model of a diode includes a voltage landmark that determines if the diode is "on" or "off". We capture landmarks using variables with the common 3-valued qualitative abstraction corresponding to the sign of the value: {Q-,Q0,Q+} with Q0 corresponding to the landmark value. The qualitative behavior of a system is a sequence of alternating intervals and instants with an instant whenever a quantity reaches a landmark. The set of all possible behaviors is called an *envisionment* and is typically represented as a graph. Each node represents a qualitative state (i.e., an assignment a qualitative value to each variable). Each edge is a successor relation that is determined by constraints on the qualitative values of their quantities.

Table 1: Relation of qualitative values for the equation $x + y = z$. "*" indicates any qualitative value.

x	y	z
Q+	Q+	Q+
Q+	Q0	Q+
Q+	Q-	*
Q0	Q+	Q+
Q0	Q0	Q0
Q0	Q-	Q-
Q-	Q+	*
Q-	Q0	Q-
Q-	Q-	Q-

2.3 Creating constraints from Modelica equations

There are a variety of methods (e.g., (Kuipers 1994)) to create constraints from simple equations. As this is not the main topic of this paper, we briefly describe our approach here. Each equation is abstracted into a relation on the qualitative values. Table 1 characterizes the relation for the equation $x + y = z$. Similar relations describe the higher-order derivatives.

To model Modelica conditional expressions, we introduce a new powerful type of *conditional* constraint. Conditional expressions change the arguments of the equation depending on the truth value of the condition. Consider equation 3 which defines a step output, y, as a function of time.

$$\texttt{y} = \texttt{offset} + (\texttt{if time} < \texttt{startTime then 0 else height}); \quad (3)$$

To create conditional constraints, it is necessary to use additional variables. We create *threshold* variables for each inequality in the condition, and a *conditional* variable for the entire conditional expression. Next, we create two conditional constraints, one for each truth value of the condition. These constraints ensure equality between the conditional variable and the appropriate clause of the conditional expression. Equation 3 results in the following constraints.

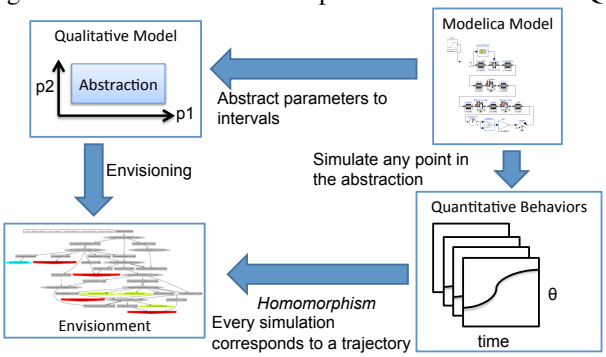
$$\texttt{threshold1} = \texttt{time} - \texttt{startTime} \quad (4)$$

$$\texttt{conditional3} = \begin{cases} 0.0 & \texttt{threshold1} < 0 \\ \texttt{height} & \texttt{threshold1} \not< 0 \end{cases} \quad (5)$$

$$\texttt{y} = \texttt{offset} + \texttt{conditional3} \quad (6)$$

The threshold variable, `threshold1`, captures the condition of the conditional expression and is defined by constraint 4. Conditional constraints (5) assigns `conditional3` to the appropriate clause of the conditional expression. Constraint 6 captures the entire original equation, equation 3. A constraint is *active* if it does not have a condition or its condition is true in the current state.

Figure 2: Soundness relationship between Modelica and QR



2.4 Sound abstraction of Modelica models

A sound abstraction of a Modelica model ensures that the results of every quantitative simulation of consistent sets of numeric parameters correspond to a trajectory in the envisionment. This relationships is shown in Figure 2. 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[2] simulation will correspond to a trajectory in the envisionment. The translation is unsound if there exists a set of valid quantitative parameters that generates a correct simulation that does not correspond to any trajectory in the envisionment.

2.5 Abstracting states

Abstracting a quantitative state into a qualitative state underlies our definition of soundness and determines the initial qualitative state for the envisionment. For each continuous Modelica variable, its sign determines its corresponding qualitative value: {Q-,Q0,Q+}. For each discrete Modelica variable, we assign its symbolic value to the qualitative variable. The remaining qualitative variables are threshold and conditional variables that correspond to combinations of Modelica variables. Therefore, we can compute their value from the quantitative state. To determine the initial qualitative state, we use OpenModelica[3] to determine the initial quantitative state and the above procedure to generate a unique qualitative state.

2.6 Abstracting behavior

Sections 2.1 and 2.2 describe how Modelica systems and qualitative systems evolve over time. In this section, we first leverage previous work to show that the continuous-time behavior of the abstracted model is correct. Then, we define our algorithm for computing qualitative effects of Modelica events and illustrate how it maintains the desired relationship between the models.

Continuous behavior In Modelica and QR, each variables' derivatives determine their continuous-time behavior. In constructing the hybrid-DAE (equations of form 1 and 2), Modelica compilers perform index reduction to arrive at an index-1 system of equations. Thus, every continuous-time variable is differentiable with respect to time, and therefore, correspond to Kuiper's *reasonable functions* (Kuipers 1994). Therefore, for any finite simulation, by the *guaranteed coverage theorem* (Kuipers 1994), the correct continuous-time behavior of the Modelica model must appear in the envisionment.

Discrete behavior Our algorithm for computing the qualitative effects of Modelica events enables the alignment of Modelica's discrete behavior (Otter, Elmqvist, and Mattsson 1999) with the envisionment. Modelica events occur when conditions change truth value. While numeric solvers must change step sizes to identify the moment that these events occur, during envisioning, events must occur at instants because the conditions are abstracted into threshold variables (as described in section 2.3). Consider `threshold1` from conditional constraints 5. The truth value of the condition changes when `threshold1` transitions from $Q+$ to $Q0$ or from $Q0$ to $Q+$.

In addition to aligning when events occur, it is necessary to align their effects. Modelica calculates the effects of events as follows. Within an event, the conditions determine the system of equations that must be solved using the state variables. If this results in changes to the truth values of the conditions, then another event is immediately triggered. The process terminates when the solution to the event system does not result in changes to the conditions.

During envisioning, an event occurs when a new state does not have the same active constraints as its predecessor (i.e., a condition changes value). We compute the qualitative results (i.e., successor states) of events using three steps: (1) identify which variables are unchanged by the event, (2) solve for all states consistent with the unchanged variables and current constraints, (3) for each new state, add it as a successor to the current state. If it does not have the same constraints as the current state, trigger another discrete event.

In step 1, we use *non-discontinuous variables* (NDV) to identify which variables are unchanged by the event. NDVs are state variables, constants, and declared by the user. We use the changing constraints to identify which variables could change at the event. We then propagate the set of potentially changing variables through the constraints stopping at NDVs. The remaining variables are unchanged through the event. This process is shown in algorithm 1[4]. First, we initialize the set of unchanged variables to all of the system variables, and the constraint queue to the symmetric difference of the previous and currently active constraints.

[2]Issues with numeric simulation algorithms may result in faulty simulation results. The envisionment may actually be used to detect such occurrences by the lack of correspondence between simulation and qualitative trajectory.

[3]www.openmodelica.org

[4]Bookkeeping that prevents constraints being added to the queue multiple times is omitted here for clarity.

Next, we loop until the constraint queue is empty. For each variable on a constraint from the queue, if it is not a state variable, we remove it from the set of unchanged variables, and add each of its constraints to the queue. In previous approaches, the computation of unchanged variables is either performed using heuristics (Nishida and Doshita 1987) or required as input (Kuipers 1994).

Algorithm 1: Compute the set of unchanged variables, UV, from the set of previous constraints, PC, active constraints, AC, and variables V. CC is the queue of constraints through which discrete changes may propagate. NDV is the set variables that are guaranteed to be maintain their values through events.

```
constantVars (PC,AC,V)
begin
    UV ← V
    CC ← (PC ∪ AC) − (PC ∩ AC)
    while notEmpty(CC) do
        c ← dequeue(CC)
        foreach v ∈ variables(c) do
            if ¬ v ∈ NDV) then
                remove v from UV
                foreach c ∈ constraints(v) do
                    enqueue(c,CC)
    return UV
```

In step 2, we generate every qualitative state consistent with the unchanged variables and the constraints from the current state. This is analogous to Modelica solving the event equation system for all the variables. In step 3, each state in this set is a successor of the current state, and if its constraints differ from the current constraints, then an additional discrete event occurs. This potential for instantaneous event sequences captures Modelica's discrete behavior.

To demonstrate the alignment of simulation results, consider the scenario of a car accelerating from rest with the parking brake on. Once the torque exceeds the static friction of the brake, the brake applies a reduced sliding friction as the wheel begins to move. The standard Modelica model represents this behavior as a discrete event when the torque exceeds static friction, followed by continuous integration to determine if the wheel starts moving forward, followed by another discrete event to change the operating mode of the brake to forward motion.

Table 2 illustrates the alignment of Modelica values with the sequence of qualitative states that occur during this scenario. The first two rows correspond to an interval state as the applied torque, `torque1.tau`, approaches the landmark representing the maximum static friction, `threshold2`. At `time` $=$ `0.25` and instantaneous state 1824, the applied torque is equal to the maximum static friction. Because any increase in `time` results in a change in the truth value for the equation that sets `startForward`, OpenModelica triggers an event. Qualitative simulation creates a continuous transition to state 2931 and identifies an event must occur because the set of active constraints has changed. The new set of constraints assign $true$ to `startForward` in state 3004. This new discrete value triggers a sequence of two more events changing the values of `locked` and the brake's acceleration. Our discrete transition algorithm terminates with state 3156 because its active constraints are the same as the previous state, 3080. The values of this state correspond directly with the values of from the Modelica simulation at `time` $=$ `0.25+`. At this point, envisioning and quantitative simulation determine that the next value of `w` is positive. This causes a change in the equations and constraints governing the `mode` variable triggering another event. This culminates in state 4098 which has corresponding qualitative values of the Modelica simulation at `time` $=$ `0.25++`, that is, the brake has just started sliding forward and is accelerating. The sequence of states from 1824 through 4098 are instantaneous from the perspective of the analysis. The final two rows of the table, `time` $= 0.252$ and `time` $=$ `0.254`, correspond with the interval state 4294.

This alignment demonstrates our algorithm for computing the qualitative effects of Modelica events. One weakness of our approach is that step 2, finding all the states consistent with the unchanged variables and current constraints, occasionally results in ambiguous branching (i.e., different sequences of instants following a event). While problematic, this set of states includes the trajectory of corresponding Modelica simulations. This ambiguity is why we allow the user to specify non-discontinuous variables. A piece of future work is to more tightly integrate our qualitative simulator with a symbolic equation solver (e.g., Macsyma (Bogen 1986)) to be able to identify unchanging variables by their equations with respect to the system's parameters and state variables.

3 Additional qualitative constraints

While the abstraction of the hybrid-DAE described in section 2.3 results in a correct envisionment, this envisionment includes many spurious states. Reducing the number of these states is necessary to provide practical feedback to designers. While unrealizable states and trajectories are a long-standing problem in qualitative simulation (Struss 1988), many states exist because the hybrid-DAE contains only the minimal set of equations necessary for numeric solvers. The second contribution of this paper is the identification of three classes of additional constraints that reduce the number of spurious trajectories without sacrificing the correctness of the qualitative model.

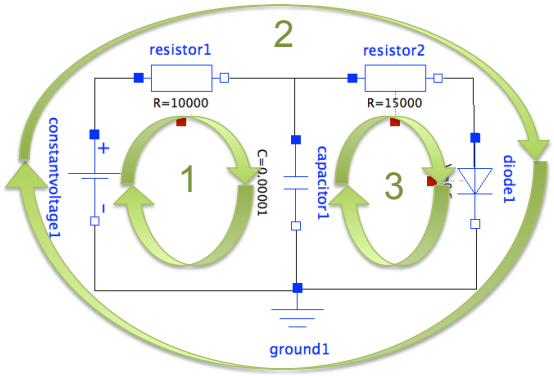
3.1 Continuity and compatibility conditions

System dynamics theorems specify the minimal set of equations necessary to enforce the continuity and compatibility conditions (e.g., Kirchoff's Voltage and Current Laws). While sufficient for Modelica simulation, this minimal set results in ambiguities for qualitative arithmetic (de Kleer and Brown 1984). Therefore, it is necessary to compute the quantitatively redundant node and loop equations by combining continuity and compatibility equations in the hybrid-DAE. We do this repeatedly adding only the equations that constrain the qualitative behavior. Figure 3 shows a Model-

Table 2: Values generated by OpenModelica and qualitative simulation of the brake model as it begins to move. Qualitative quantity values are given for the angular velocity and applied torque with respect to the maximum static friction, `threshold2` = `torque1.tau` − 0.25. The `state` column provides identifiers for the corresponding qualitative state. '*' indicates that the qualitative state was created by the discrete event algorithm and not continuous integration. '+' indicates that the value has been truncated for presentation, and the real value is slightly greater than the value in the field. '—' is used for fields with no corresponding Modelica values. The headings `w`, and `a` correspond to the speed, and acceleration respectively of the brake. `brake1.tau` is the force applied by the brake against the applied torque of the system, `torque1.tau`.

time	state	w , QValue(w)	a	brake1.tau	torque1.tau , QValue(threshold2)	locked	start Forward	mode
0.246	1255	0.0 , [Q0,→]	0.0	0.246	0.246 , [Q-,↑]	true	false	stuck
0.248	1255	0.0 , [Q0,→]	0.0	0.248	0.248 , [Q-,↑]	true	false	stuck
0.25	1824	0.0 , [Q0,→]	0.0	0.25	0.25 , [**Q0**,↑]	true	false	stuck
—	2931	— , [Q0,→]	—	—	— , [**Q+**,↑]	true	false	stuck
—	3004*	— , [Q0,→]	—	—	— , [Q+,↑]	true	**true**	stuck
—	3080*	— , [Q0,→]	—	—	— , [Q+,↑]	**false**	true	stuck
0.25+	3156*	0.0 , [Q0,↑]	**0.125+**	**0.125**	**0.25+** , [Q+,↑]	false	true	stuck
—	3753	— , [**Q+**,↑]	—	—	— , [Q+,↑]	false	true	stuck
—	3949*	— , [Q+,↑]	—	—	— , [Q+,↑]	false	true	**forward**
—	4022*	— , [Q+,↑]	—	—	— , [Q+,↑]	false	**false**	forward
0.25++	4098*	1.2E-10 , [Q+,↑]	0.125+	0.125	0.25+ , [Q+,↑]	false	false	forward
0.252	4294	0.00025 , [Q+,↑]	0.127	0.125	0.27 , [Q+,↑]	false	false	forward
0.254	4294	0.00051 , [Q+,↑]	0.129	0.125	0.29 , [Q+,↑]	false	false	forward

Figure 3: Modelica model of a diode in parallel with a capacitor annotated with the three KVL loops.



ica model for a diode and capacitor in parallel. The hybrid-DAE includes equations 7 and 8 for voltage loops 1 and 2:

$$\texttt{r1.v} = \texttt{bat1.V} - \texttt{c1.v}; \quad (7)$$

$$\texttt{r1.v} = \texttt{bat1.V} - \texttt{r2.v} - \texttt{d1.v}; \quad (8)$$

By combining equations 7 and 8, we arrive at the equation for the third voltage loop.

$$\texttt{r2.v} = \texttt{c1.v} - \texttt{d1.v}; \quad (9)$$

While equation 9 is redundant from the perspective of Modelica simulation, the resulting constraint rules out additional qualitative states. For example, the following values, `bat1.v` = Q+, `r1.v` = Q+, `r2.v` = Q+, `d1.v` = Q+, and `c1.v` = Q-, are consistent with the constraints from equations 7 and 8, but are ruled out by the constraints resulting from equation 9. Thus, we add this equation to the system before creating the constraints.

3.2 Higher-order derivative equalities

Another technique for generating additional constraints concerns equalities between variables with explicit derivatives. Consider the equations 10-12 modeling a brake attached to a flywheel. While their positions and velocities are equal, the hybrid-DAE represents this in three equations.

$$\texttt{brake1.phi} = \texttt{flywheel1.phi}; \quad (10)$$

$$\texttt{brake1.w} = \texttt{der(brake1.phi)}; \quad (11)$$

$$\texttt{flywheel1.w} = \texttt{der(flywheel1.phi)}; \quad (12)$$

Converting these three equations to qualitative constraints fails to capture the equality between the velocities. Therefore, for any two variables that are equal and have explicit derivatives, we add equality constraints between their derivatives. In this case, we add equation 13 ensuring that the velocity of the flywheel is equal to the velocity of the brake.

$$\texttt{brake1.w} = \texttt{flywheel1.w} \quad (13)$$

3.3 Partially ordered landmarks

The hybrid-DAE does not explicitly define *quantity spaces*, or the important values for each variable. Threshold variables must be computed from the conditional expressions. Consider the threshold variables defined by equations 14 and 15. `brake1.sa` is a quantity representing the torque applied by the brake while it is stuck. `brake1.tau0_max` and `brake1.tau0` are parameters and correspond to the maximum static friction and initial sliding friction respectively.

$$\texttt{threshold3} = \texttt{brake1.sa} - \texttt{brake1.tau0} \quad (14)$$

$$\texttt{threshold4} = \texttt{brake1.sa} - \texttt{brake1.tau0\_max} \quad (15)$$

Consider a Modelica model where $\texttt{brake1.tau0} = 0.25$ and $\texttt{brake1.tau0\_max} = 0.5$. In the qualitative state where $\texttt{brake1.sa} < \texttt{brake1.tau0}$ and increasing, threshold3 and threshold4 have the same qualitative value, [Q-:↑]. That is, the quantity brake1.sa is approach both thresholds. Envisioning will consider four cases for the successor states: (1) threshold3 is reached, (2) threshold4 is reached, (3) threshold3 and threshold4 are reached at the same time, and (4) neither threshold is reached. Using just the constraints resulting from equations 14 and 15, none of these situations can be ruled out. Because $\texttt{brake1.tau0} < \texttt{brake1.tau0\_max}$, we create constraint 16 to capture the partial ordering of the thresholds. Thus, if a threshold is reached, it will be threshold3.

$$\text{Q+} = \texttt{threshold3} - \texttt{threshold4} \quad (16)$$

3.4 Case study

To illustrate the value of these additional constraints, we conducted the following case study. We created three simple systems (described below) built from electrical and mechanical components from the Modelica Standard Library [5]. We performed two envisionments of each model: a baseline without the additional constraints and one with all of the additional constraints. We measure the number of qualitative states produced by the envisionment and expect a reduction in size in the additional constraints case.

- Brake: A ramp source torque attached to a brake applied to a stationary flywheel.
- RC-ladder: A battery is connected a ladder of three resistors and three capacitors
- Adby: An oscillatory circuit consisting of two resistors, two capacitors, and an inductor

Table 3: Number of qualitative states resulting from a simulation with different classes of constraints.

Model	Baseline	Additional Constraints
Brake	85	37
RC-ladder	241	28
Adby	5858	646

In each of these systems, we observed a significant reduction in the number of qualitative states. Because the additional constraints act to filter unrealizable behavior, every state in the reduced envisionment also appears in the baseline envisionment. We expected this reduction to be exponential. Therefore, we performed the experiment on a sequence of five RC-ladders of increasing size and found that the log of number of states filtered by additional constraints has a linear relationship with the number of components in the model. Therefore, the reduction is exponential.

[5] https://www.modelica.org/libraries

4 Related work

While numerous approaches to qualitative simulation have addressed issues surrounding discrete events, they all require additional modeling than what is contained in Modelica's hybrid-DAE. For example, QSIM (Kuipers 1994) requires a user-supplied transition mapping function when there is a change in operating regions. This function includes the new set of constraints, the variables whose magnitudes are unchanged, the variables whose derivatives are unchanged, and any values that must be asserted. These system-level events are not compositional. De Kleer and Brown use *modes* and propagate non-local discrete changes through heuristics to provide a causal account of device behavior (de Kleer and Brown 1984). Our algorithm determines the qualitative effects of declarative events from component models. Nishida and Doshita present two methods for modeling discrete changes in envisioning: (1) as continuous change that happens over infinitesimals, (2) as a sequence of mythical instants which may be inconsistent with the model (Nishida and Doshita 1987). Iwasaki *et al.* formulate instantaneous discrete changes using rules and hyperreal time semantics (Iwasaki et al. 1995). Our approach draws on the idea of infinitesimals and mythical instants that may be inconsistent with respect to the constraints to define our instantaneous event sequences. These ideas are essential to capturing Modelica's discrete behavior in the envisionment.

5 Discussion

The purpose of this work is to apply qualitative reasoning to simulation models without burdening the designer. We define a sound and effective mapping between Modelica and our qualitative modeling approach. To accomplish this, we introduce an algorithm that determines the qualitative effects of events directly from the acausal Modelica model. Declarative events enable physical components models to be reused in many contexts. While our approach does not employ heuristics or require the additional modeling knowledge like previous approaches, we do allow the user to specify pseudo-state variables that maintain their values through discrete transitions, to reduce unrealizable trajectories. Directly translating the hybrid discrete and algebraic equations from Modelica into constraints results in an envisionment with many unrealizable trajectories. By making the implicit information of equations explicit, we reduce the set of unrealizable trajectories improving the utility of the resulting envisionment. We identify three classes of quantitatively redundant relations in the hybrid-DAE: (1) the continuity and compatibility relations, (2) relations between higher-order derivatives, and (3) qualitative landmark ordering relations. We show that these constraints exponentially reduce the size of the envisionment.

These contributions are an important step toward integrating qualitative reasoning into design tools. Automating qualitative reasoning could free the designer to focus on more creative parts of the design process as well as new techniques for design space exploration and guided quantitative analysis.

Acknowledgments

This work was partially sponsored by The Defense Advanced Research Agency (DARPA) Tactical Technology Office (TTO) under the META program and is Approved for Public Release, Distribution Unlimited. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

References

Bogen, R. 1986. *MACSYMA reference manual*. Symbolics, Incorporated.

de Kleer, J., and Brown, J. S. 1984. A qualitative physics based on confluences. *Artificial Intelligence* 24(1):7–84. Also in: Bobrow, D. (ed.) *Qualitative Reasoning about Physical Systems* (North-Holland, Amsterdam, 1984 / MIT Press, Cambridge, Mass., 1985).

Forbus, K. D. 1984. Qualitative process theory. *Artificial Intelligence* 24(1):85–168. Also in: Bobrow, D. (ed.) *Qualitative Reasoning about Physical Systems* (North-Holland, Amsterdam, 1984 / MIT Press, Cambridge, Mass., 1985).

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

Iwasaki, Y.; Farquhar, A.; Saraswat, V.; Bobrow, D.; and Gupta, V. 1995. Modeling time in hybrid systems: How fast is" instantaneous"? In *International Joint Conference on Artificial Intelligence*, volume 14, 1773–1781. Citeseer.

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

Lattmann, Z.; Pop, A.; de Kleer, J.; Fritzson, P.; Janssen, B.; Neema, S.; Bapty, T.; Koutsoukos, X.; Klenk, M.; Bobrow, D.; Saha, B.; and Kurtoglu, T. 2014. Verification and design exploration through meta tool integration with openmodelica. In *Proceedings of the 10th International Modelica Conference*.

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

Otter, M.; Elmqvist, H.; and Mattsson, S. E. 1999. Hybrid modeling in modelica based on the synchronous data flow principle. In *Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on*, 151–157. IEEE.

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

Simko, G.; Levendovszky, T.; Neema, S.; Jackson, E.; Bapty, T.; Porter, J.; and Sztipanovits, J. 2012. Foundation for model integration: Semantic backplane. In *Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE*, 12–15.

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

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