Placing Qualitative Reasoning in the Design Process

Matthew Klenk, Daniel G. Bobrow, Johan de Kleer, John Hanley, Bill Janssen
Palo Alto Research Center
3333 Coyote Hill Rd
Palo Alto, CA, 94304
{klenk,bobrow,dekleer,hanley,janssen}@parc.com

Abstract

Design of physical devices to meet requirements involves exploration of alternative configurations, and evaluation of performance and tradeoffs. In addition to numeric analyses and quantitative simulations, designers use qualitative knowledge about components, devices, and contexts of use to arrive at suitable designs. To put qualitative reasoning into the hands of the designers, it is necessary to integrate with existing design tools (e.g., OpenModelica) in a *design exploration environment*. This environment provides access to standard model libraries, and numeric simulation of fully specified designs. Before selecting all parameters of a design, the integrated qualitative verification we describe can be used to determine if the design will meet requirements for all, some, or no set of numeric parameters. It can also identify derived requirements. Qualitative verification can also be used to support design space exploration through a tool that automatically generates candidate designs and identifies promising designs. To support detailed design, we demonstrate the automatic extraction of *guards*, parameter inequalities that focus the simulation toward desired outcomes.

Qualitative Reasoning and Design

The design of physical devices is a core task for professional engineers in the automotive, aerospace, and electronics industries. With crowd sourcing, design of such devices is now also undertaken by amateurs around the world. An example is the team at Wikispeed[1], which is designing a 100 mpg car in their free time. Improving available tools by adding qualitative analysis should not only reduce the amount of time required to create innovative solutions, but also improve the final design.

Qualitative reasoning has its roots in automating reasoning about physical systems (Forbus 1984; de Kleer & Williams 1992). Based on the intuition that engineers employ qualitative reasoning extensively throughout the design process, numerous researchers have sought to exploit qualitative reasoning in the context of design (Franke 1991; Iwasaki 1997; Everett 1999; Wetzel & Forbus 2009). While producing advances in qualitative reasoning, these approaches have fallen short of affecting the design process in practice because they have not worked well with existing design tools, a notable exception is the AutoSteve (Struss & Price 2003). Our goal is to build a better *design exploration environment* by integrating qualitative analysis with existing software tools and libraries for graphical construction and simulation of designs. Therefore, we hope to support designers with qualitative reasoning techniques enabling them to identify more potential pitfalls, consider a broader range of designs, and guide detailed design.

Verification is the process of proving a design meets certain requirements. Given a design with unspecified parameters and a context of use, *qualitative verification* performs qualitative simulation to determine if the design meets a set of requirements for all, some, or no choices of component parameters. In this paper, we show how the results of qualitative verification support automated design space exploration and detailed design. By filtering automatically generated topologies qualitatively, we reduce the space of topologies under consideration. When qualitative verification indicates some possible failures, and some potentially successful designs, we use the envisionment to infer parameter inequalities, or *guards,* that prevent the failure behaviors from occurring. These guards provide guidance for a more fully specified design. For example, a design of an electrical circuit may meet requirements only when the voltage of a power source is greater than the turn on voltage of a diode.

This paper is organized as follows. In the next section, we discuss our efforts in creating a design exploration environment by integrating with OpenModelica, an open source simulation system that supports the standard modeling language, Modelica. This is followed by a discussion of qualitative verification and an illustrative example identifying potential design problems. Next, we present an automatic design space exploration system that uses qualitative verification to filter topologies. This is followed by a discussion of our approach for automatically

[1] wikispeed.com

extracting guards from feasible designs. We close with a discussion of related and future work.

Design Exploration Environment

Designers currently use sophisticated tools for creating and analyzing designs. To make qualitative analysis useful to these designers, it needs to be easily accessible from a standard tool. Consequently, we have been integrating qualitative reasoning into OpenModelica[2] (Fritzson 2004), an open source Modelica modeling and simulation environment. OpenModelica provides a graphical model editor (shown in Figure 1) that is familiar to designers as well as access to the Modelica Standard Library with over 1200 models.

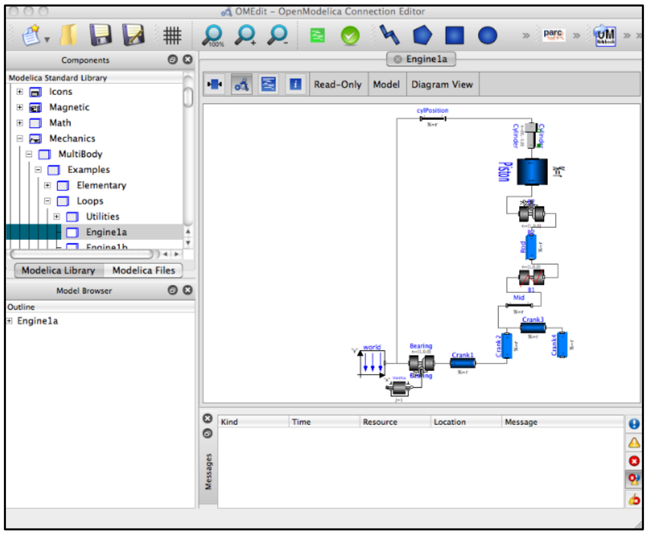

Figure 1: OpenModelica graphical connection editor

Designers use OpenModelica to create models and perform simulations. Models are created by selecting components from the standard model library, and creating connections between them. This is identical to the process used in creating qualitative component models. Component models in Modelica are defined by sets of ordinary differential equations (ODEs) describing the constraints among the parts of the components. Qualitative models will have the same structure, except that they use qualitative differential equations (QDEs) (Kuipers 1994) to describe component behavior.

Both OpenModelica and qualitative simulators flatten the component models to perform the simulation – that is eliminate component boundaries, and create a single set of (potentially optimized) equations that are then handled by the simulation engine. For Modelica, this is a numeric simulation. For qualitative simulation, the equations of transformed into a set of constraints that determine the temporal evolution of the system.

These parallels in system structure suggested two approaches to us for integrating qualitative reasoning with OpenModelica: (1) abstract component models, and (2) extract equations produced by the OpenModelica compiler. We describe each in turn.

[2] www.openmodelica.org

Text-to-Text Translation of Component Models

Modelica defines declarative, acausal, discrete-continuous hybrid models making it an appropriate input language for qualitative simulation. To facilitate our ongoing automatic translation efforts, we aligned our model construction language (QML) with Modelica (see the corresponding models in Figure 2). The composition of models occurs through connections that are domain specific (e.g., electrical pins defined in the one port model). While each model defines variables `i` and `v`, the Modelica variables have ranges of real numbers, and the qualitative variables have finite ranges defined by the *quantity space* for each variable (Kuipers 1994). In Modelica, these variables are differentiable and in our qualitative system each value always has a qualitative derivative, or direction (e.g., increasing, steady or decreasing). These derivatives enable their corresponding simulation engines to generate the behavior of the system (Williams 1984). In each modeling language, the composition of the models creates additional constraints on the flow and effort variables of the models governed by Kirchhoff's laws.

Modelica	QML
`model Capacitor`	`(defprototype Capacitor`
`extends onePort;`	`:extends onePort`
`parameter Capacitance C;`	`:parameters ((C))`
`equation`	`:equations`
`i = C * der(v);`	`((= i`
`end Capacitor`	`(* C (deriv v 1)))))`

Figure 2: Qualitative capacitor model is a simple transformation of the one in the Modelica Standard Library

One area where we differ from Modelica representation concerns our use of modes instead of conditional equations. Modes offer the following advantages: (1) they localize the definition of hybrid behavior for the component, and (2) they provide a natural way to model various faulty behaviors.

```
(defprototype ideal-diode :extends (one-port)
   :variables ((v voltage :landmarks
                          (Q0 OnVoltage)))
   :mode (off :entry ((= i Q0))
              :equations ((= i Q0)))
   :mode (on :entry ((= v OnVoltage))
             :equations ((= v OnVoltage))))
```

Figure 3: Diode model with two modes

To illustrate our modeling approach, Figure 3 contains our definition of an ideal diode. We present this here in our internal S-expression syntax, which highlights this localization.[3] This model is a subclass of the electrical one port model, which defines two electrical connections, a positive pin and a negative pin, and variables for the current and voltage of the diode. The redefinition of the voltage variable v is essential to create the quantity space including `Q0`, representing 0V, and `OnVoltage`, representing the turn on voltage for the diode. The diode has two modes, `off` and `on`. The component is in a mode until the entry conditions for another mode have been satisfied, in this case, if the diode was `off`, the equation stating that no current passes through the diode is enforced. This persists until the instant when the voltage across the diode equals `OnVoltage`, at which point the equation holding the voltage constant is enforced, and current is no longer constrained and can flow through the diode.

Using OpenModelica Compilation

In order for OpenModelica to perform numeric simulation, it generates a single set of quantitative equations. As an alternative to translating the component based model, we have been exploring creating the qualitative constraints for qualitative simulation by abstracting the equations constructed by the OpenModelica Compiler. Using this approach has two advantages over text-to-text transfer. First, Modelica includes language constructs for supporting complex model construction in its models, For example, one can create a transformer with a parameter-defined number of inductors. We did not want to support the text-to-text interpretation of these constructs. The OpenModelica compiler handles these as part of its flattening process.. Second, the OpenModelica Compiler does both algebraic simplification and index reduction as part of the compilation process. Given the set of equations produced by the OM compiler, we are often able to create the set of constraints necessary to perform qualitative simulation. There are still certain Modelica constructs that we cannot handle – but we are gradually reducing the size of this untranslatable set. So far, we have translated equations from models with only continuous dynamics such as RLC oscillators.

Each of these approaches faces difficulties in the breadth of operators available in Modelica. For example, Modelica models may define arbitrary algorithms for computing particular outputs. An important part of our future work is systematically identifying which Modelica constructs can be translated into qualitative models and understanding how to treat those that can not.

[3]We have defined a standard template for expressing modes that is acceptable to current Modelica compilers, but do not include it here.

Qualitative Verification

Qualitative simulation is possible very early in the design process, before parameters are selected, when alternative system topologies are being considered. Given a qualitative model, an initial state, and a set of requirements, qualitative verification determines if the design will satisfy the requirements. In this section, we describe our qualitative simulation algorithm, how we incorporate requirements into the simulation, and provide an example of qualitative verification using a vehicle ramp door.

Simulation

Qualitative simulation (de Kleer and Williams 1992), or envisioning, is the process of projecting forward, from an initial situation and a model, all possible qualitative states that may occur. Our qualitative simulation algorithm inherits largely from QSIM (Kuipers 1994). Given a *state* (an assignment of qualitative values to each variable), qualitative simulation computes possible successor values for each variable and uses constraints to determine combinations which make up consistent next state(s). Consider a position variable that was between the `Closed` and `Open` landmarks and moving toward `Closed`. There are four possible successor values for this variable. Its value may remain in the interval or reach the `Closed` landmark and it may continue increasing or become steady (its derivative stays positive or becomes 0).

Cyber-physical systems include dynamics that are discrete as well as continuous (e.g., an input signal to open the door, the changing of gears in a drive train, a diode switching from off to on). We model such changes through *modes*, which include an entry condition, new values for some variables, and equations that are valid within that mode. During simulation, discrete changes occur at instants when mode entry conditions are satisfied. The initial values and equations govern the behavior of quantities in the following interval. The result of qualitative simulation is a directed graph of states.

Requirements

Cyber-physical systems (CPS) need to satisfy many requirements. *Static requirements* deal with the structure of the object (e.g., a car must weigh less than 2500kg), and can in general computed from the model. *Behavioral requirements* deal with the response of the object during a use case, or scenario. In general, these need to be verified through simulation. Behavioral requirements may concern a prohibited state of the system (e.g., where the engine RPM exceeds its redline value) or undesirable sequences of states during a trajectory (e.g., while traveling on level terrain at constant velocity, the automatic transmission should not chatter between gears). We use linear temporal logic as a formal language for stating these requirements on trajectories (Emerson 1990). If these requirements are

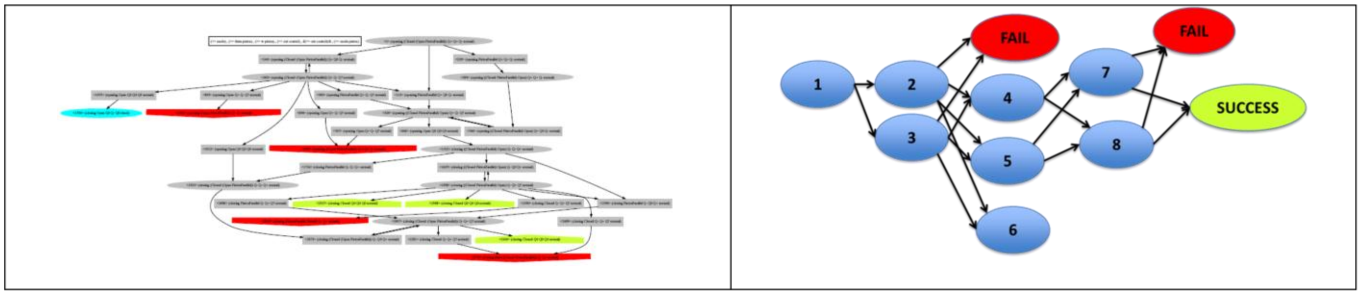

Figure 4: The envisionment graph for the door system model produced by our system (left) and a simplified version for presentation (right)

supplied to our qualitative simulation algorithm as success and failure conditions, we terminate a trajectory in the simulation if the condition is met. Because requirements are composable (e.g., every engine has a redline speed which should not be exceeded), it is possible to model requirements as components within the system. Requirement components include mode transitions to modes named `success` or `failure` signaling the envisionment algorithm to terminate the behavior.

After the envisionment graph has been created, qualitative verification provides the following analysis. If none of the trajectories violate requirements, then for all consistent assignments of parameters, the system will satisfy all requirements. That is, the system will not reach a state that violates a safety requirement or transition along an undesirable trajectory. If some trajectories violate requirements and others do not, then the design may satisfy the requirements with appropriate constraints on component parameter values. In either case, detailed design is required to determine the assignment of parameter values that best fits the needs of the designer. If all of the trajectories violate requirements, detailed design is not necessary because no set of parameter values will satisfy the requirement.

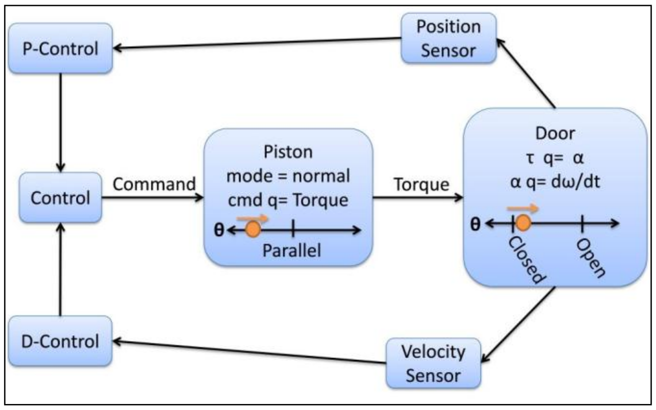

Figure 5: Architectural of the door system

Verification Example: Vehicle door linkage

To illustrate qualitative verification, consider the door system shown in Figure 5. The architectural model shows quantity spaces for the positions of the piston that moves the door, and the door itself. The system consists of a PD controller, which uses position and velocity sensors from the door; a piston, whose linear motion applies a torque on the door; and the door slab itself, that rotates around a bottom hinge. An input signal to the controller specifies the desired next position for the door. This door angle has two landmarks in the position quantity space, `Closed` and `Open`, and the piston has one landmark on the linear position quantity space, `parallel`, representing the angle where the piston acts in parallel with the hinge. We will evaluate this design against the requirements that the door should always be able to be closed, and the door position should always be between the door open and door closed position inclusively. As a use case, we use a scenario in which the door starts closed, and (1) the command is given to open the door, and (2) when the door has reached the open position, the command will be given to close the door.

Our system produces the envisionment shown in Figure 4. The left pane shows the actual envisionment, indicating that even for this simple system, there are many states and trajectories. The simplified graph in the right pane illustrates what the designer can learn from the envisionment. In this use case, the design may reach a successful situation (green node labeled SUCCESS). Red nodes in the graph are qualitative states that violate requirements. Appropriate parametric assignment will be needed to ensure that trajectory for each failed state is avoided.

Further analysis of the envisionment provides additional guidance for detailed design. There is a terminal situation, (node 6 in the simplified graph, cyan in the full graph), that does not satisfy either the success or failure conditions of the system. In the example, this state results from a kinematic singularity in the piston door connection. That is, when the acting angle of the piston is parallel to the angle of the door, the piston produces no torque. While this is part of the piston component model, it only leads to a quiescent (terminal) state if the door is stationary at this point. This unmarked dead-end state suggests the need for an additional requirement to guide the designer to avoid this state. This risk case would not become apparent in a simpler use case; it requires one where the door first opened and then closed. This indicates the need for careful design of use cases.

Automated Design Space Exploration

Innovative design often requires a search for different configurations of existing components (new topologies) to achieve a specified functionality. This search space is exponential in the number of components in the design. Qualitative verification can help prune the design space in two ways. By using qualitative models of components, each qualitative component model corresponds to a many possible quantitative components. Therefore, one simulation can summarize many numerical experiments. Secondly, when a qualitative simulation shows that no choice of parameters will satisfy the requirements, that topology can be eliminated from the search space. If there is a feasible design, the envisionment can guide parameter selection in a detailed design.

As an example of design space exploration, consider a requirement for a system that turns on a light emitting diode a short but perceptible time after a switch is turned on. The available components include batteries, switches, resistors, capacitors, inductors, and diodes. The topological design space includes every configuration of these components. To illustrate the utility of qualitative verification, we describe a design space exploration tool that searches the design space by taking one of the following design actions: adding a component in parallel or series with an existing component, removing a component, or flipping a component in the circuit. Figure 6 illustrates the starting design, which includes just a battery, switch and diode.

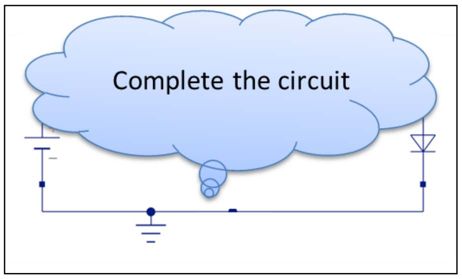

Figure 6: Starting point for topological design space exploration

After each design action, we build a qualitative model and simulation for the current design candidate. Many candidates are equivalent to shorted or open circuits. Our system identifies them because their initial conditions are inconsistent. If the design candidate has consistent initial conditions, our system performs qualitative verification. Consider a circuit with a resistor completing the circuit in Figure 6. The envisionment of this will begin with both the switch and diode off. It has two trajectories following the instant the switch is turned on. In one, the diode is on, and, in the other, the diode is off. The trajectory of the actual system depends on the ordinal relationship between the on voltage for the diode and the battery's voltage. Because neither of these trajectories satisfies the requirement that there exists a delay before the light turns on, qualitative verification eliminates this topology without considering all possible combinations of battery voltages, resistances and on voltages.

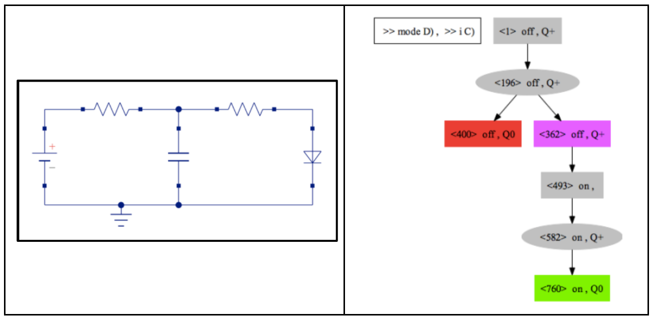

Figure 7: The envisionment on the right proves that the topology on the left can satisfy the requirements

Now consider the design in Figure 7. QRM produces an envisionment with two trajectories. They are identical in the interval after the switch is turned on, the capacitor is charging and the voltage across the diode is increasing. This interval terminates in one of two instants: (1) the current ceases flowing into the capacitor and the system reaches a steady state, and (2) the voltage across the diode reaches the on voltage landmark causing a mode transition (shown in magenta) resulting in the diode turning on. This second trajectory satisfies the requirement. Therefore, this topology is a candidate design. In the next section, we describe a method for generating constraints to guide detailed design.

Focusing the Envisionment with Guards

For many designs, qualitative verification will result in both trajectories that satisfy the requirements and trajectories that do not. As observed by Iwasaki (1997) this ambiguity is useful in design because it can alert designers to potential problems, such as the kinematic singularity in Figure 4. It can also be used to construct additional constraints to guide detailed design. In this section we present *guards*, which are constraints on the model that focus the envisionment toward successful states.

One source of ambiguity arises from the fact that each quantity space is defined with respect to a single component. Consider the feasible diode circuit in Figure 7.

In this model, there is a battery voltage landmark in the battery model and turn on voltage landmark in the diode model. The ambiguity in the simulation results from ambiguity in the ordering of these landmarks. However, determining properties to ensure the correct trajectory is not straightforward. We propose a method for deriving inequalities through model construction and qualitative verification.

Our algorithm is as follows. First, we propagate quantity spaces across constraints. Consider the constraint specifying the voltage across the battery, v = p.v – n.v. The only variable with a landmark other than 0 is the variable v has a positive landmark `VBat`. We generate a symmetric quantity space, one in which the inverse of each landmark is included (e.g., `(-∞, -VBat, Q0, VBat, ∞)`) for each variable. We do this for each non-simple quantity space (i.e., with landmarks other than 0). Next, we look for each variable with multiple quantity spaces and create a set of single quantity spaces representing a total ordering of the variables. In our diode example, each voltage variable will have two quantity spaces, one with the `VBat` landmark and one with `OnV` landmark. Therefore, we generate three quantity spaces representing the possible orderings the two landmarks:

1. `OnV > VBat (-∞, -OnV, -VBat, Q0, VBat, OnV, ∞)`
2. `OnV = VBat (-∞, -L1, Q0, L1, ∞)`
3. `OnV < VBat (-∞, -VBat, -OnV, Q0, OnV, VBat, ∞)`

`L1` is a new landmark that is created that is equal to `OnV` and `VBat`. We create three qualitative models with the same set of constraints, but with different quantity spaces for the voltage variables in the model.

We perform qualitative verification on each of the systems. The envisionment of the first system consists of three states terminating when the capacitor is charged and the diode is off. By simulating the second system, our system determines that the initial conditions are inconsistent because the only trajectory leads to an endless loop of mode transitions (i.e., the diode switches between off and on). The third system results in an envisonment of twelve states in which all trajectories satisfy the requirements. Therefore any parameter settings that satisfy the inequality defining the system, `OnV < VBat`, will result in a system that meets the requirements. In this manner, we can infer inequalities to guide parameter selection in detailed design.

Future Work

This work is part of the METAX effort whose goal is a 5x reduction in the amount of time required to design and verify complex cyber physical systems (e.g., satellites, aircraft, and military ground vehicles). These systems have thousands of components and equations. To support such designs we need both modeling breadth and scalability. By breadth here, we mean the ability to import and abstract an appropriate significant portion of the Modelica Standard Library. The scale of qualitative simulation needed is a second challenge.

A challenge with respect to breadth is the useful expressiveness of the Modelica modeling language, especially as it has been used as a general programming language. Modelica includes many constructs, in addition to equations and discrete variables, that are sometimes used in modeling components. One example is the user-defined algorithm. While some algorithms have straightforward qualitative analogs (e.g., linear interpolation tables), others involve complex operations (e.g., quaternion for spatial rotations). We intend to systematically analyze the Modelica Standard Library to determine what portion may be directly imported to create large qualitative models.

Given a large qualitative model, scale is a challenge in performing qualitative simulation. Factoring, or decomposition, has long been an important way of scaling up qualitative simulation. This involves simulating subcomponents individually and maintaining a record of their mutual constraints. A common approach is to divide the system at component boundaries (Clancy & Kuipers 1997). Factoring the model by spatio-temporal interactions has been useful for performing battlespace simulations (de Kleer *et al.* 2009). Guglielmann & Ironi (2010) present a decomposition method by analyzing the causal ordering of the equations. Our largest simulation to date involves a drivetrain model with 61 variables and results in 12896 situations. In addition to implementing existing factoring techniques, we believe we will need new ideas to simulate and verify systems at the scale of a full military vehicle.

Related Work

In addition to the qualitative reasoning work mentioned throughout this paper, Shults and Kuipers (1997) present the first qualitative verification system, which proves properties about continuous systems expressed in computational tree logic (Emerson 1990). Struss and Price (2003) report on applications of qualitative reasoning in automotive design focusing on diagnosis. Our work hopes to build on these results by placing qualitative reasoning into OpenModelica, a popular open source design tool. Sacenbacher and Struss (2005) provide a detailed analysis of the problems posed by locally defined quantity spaces. Our work departs from theirs by including trajectory requirements necessitating simulation.

Alternative formulations of this problem arise from the fields of hybrid systems and model checking. Hybrid systems require quantitative models and numerical parameters. Such information is often not available in early design. In contrast to our approach of constructing a model from components, HybridSAL (Tiwari 2008), a hybrid system verification system, begins with a set of equations, with numeric parameters chosen. While HybridSAL has the advantage of being able to answer quantitative questions about a design (e.g., will the vehicle reach 30 mph in 6 seconds), it is limited to linear models. Other researchers have explored the use of the PRISM model checking system (Kwiatkowska *et al.* 2011) to perform verification of cyber-physical systems. PRISM models have the advantage that they can consider probabilistic state transitions. Probabilistic state transitions make PRISM particularly useful for verifying requirements about the likely reliability of systems given failure rates of components (e.g, "what is the probability that vehicle will be able to operate continuously for 570 hours"). A challenge for doing this analysis is that there is no automatic way to move from equations specifying components to the models used by PRISM.

Discussion

While the design of physical systems is a well-known application area for qualitative reasoning, there have been few efforts to integrate QR into existing modeling and simulation tools. Our design exploration environment seeks to enable designers to use a large existing library of 1200+ components within a familiar interface. With requirements specified in temporal logic, we use qualitative verification to give following feedback: (1) whether the design work for all, some, or none of the parameter values, (2) which requirements may be violated, and (3) what derived requirements are needed to avoid dead-end states. We also showed how qualitative verification can be used within an automated design space exploration system to identify feasible component topologies. Finally, we described how parameter constraints, or guards, can be inferred by performing qualitative verification on multiple models.

While we face challenges with respect to importing models and scaling up, the above examples illustrate the promise of our approach. Success in this research project will place the power of qualitative reasoning in the hands of expert and non-expert designers alike thereby reducing the amount of time required to design complex cyber-physical systems.

Acknowledgements

We thank Peter Bunus (Linkoping University) for helping us understand the nuances of Modelica models, Christian Fritz for his help generating candidate designs, and Peter Struss for discussions pertaining to qualitative verification. This work was 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

Brajnik, G. and Clancy, D. 1996. Trajectory constraints in qualitative simulation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-96)*, AAAI/MIT Press, 1996.

Clancy, D. and Kuipers, B. 1994. Model decomposition and simulation. In *Working Papers of the Eighth International Workshop on Qualitative Reasoning about Physical Systems (QR-94)*, Nara, Japan

deKleer, J., Forbus, K., Hinrichs, T., Sungwook, Y., and Jones, E.K., 2009. Factored Envisioning. In *Proceedings of the 23rd Annual Workshop on Qualitative Reasoning*, Ljubljana, Slovenia.

de Kleer, J. and Williams, B.C. 1992. Special volume on Qualitative Reasoning about Physical Systems II, *Artificial Intelligence*. Elsevier.

Emerson, E. 1990. Temporal and Modal Logic. In van Leeuwen, J. ed. *Handbook of Theoretical Computer Science.* North Holland: Amsterdam.

Everett, J. O. 1999. Topological inference of teleology: Deriving function from structure via evidential reasoning. *Artificial Intelligence*, 113 (1-2).

Franke, D. 1991. Deriving and using descriptions of purpose. *IEEE Expert*, April 1991, pp. 41-47.

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

Forbus, K. 1984. Qualitative process theory. *Artificial Intelligence*, 24. Elsevier 85-168.

Guglielmann, R. & Ironi, L. 2010. A divide-and-conquer strategy for qualitative simulation of complex dynamical systems. In *Proceedings of the 24th Annual Workshop on Qualitative Reasoning*. Portland, Oregon.

Iwasaki, Y. 1997. Qualitative reasoning and the sciences of design. *IEEE Expert: Intelligence Systems*. Special issue on AI in Design. D. Brown and W. Birmingham, eds. April.

Kuipers, B. 1994. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. Cambridge, MA: MIT Press.

Kwiatkowska, M., Norman, G., and Parker, D. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. *In Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of LNCS, pages 585-591, Springer

Sachenbacher, M. & Struss, P. 2005. Task-dependent qualitative domain abstraction. *Artificial Intelligence* 162 (2005) pp. 121-143 ISSN: 0004-3702.

Shults, B. and Kuipers, B. 1997. Proving properties of continuous systems: qualitative simulation and temporal logic. *Artificial Intelligence*. 92: 91-129, 1997.

Struss, P. and Price, C. 2003. Model-based systems in the automotive industry. *AI Magazine*. 24(4). AAAI Press.

Tiwari, A. 2008. Abstractions for hybrid systems. *Formal Methods in Systems Design*, 32:57–83.

Wetzel, J. and Forbus, K. (2009). Automated Critique of Sketched Mechanisms. *Proceedings of the 21st Innovative Applications of Artificial Intelligence Conference*. Pasadena, California.

Williams, B. 1984. The Use of Continuity in Qualitative Physics. In *Proceedings of the National Conference on Artificial Intelligence*, Austin, TX, pp. 350-354.