In *The Eleventh International Workshop on Qualitative Reasoning (QR-97)*
Cortona, Italy, June 1997.

Model Revision: Techniques and tools for analyzing simulation results and revising qualitative models

Daniel J. Clancy[1], Giorgio Brajnik[2], and Herbert Kay[1]

[1]Department of Computer Sciences
University of Texas at Austin, Texas 78712
clancy@cs.utexas.edu, bert@cs.utexas.edu

[2]Dip. di Matematica e Informatica
Università di Udine, Italy
giorgio@dimi.uniud.it

Abstract

One of the factors hindering the wide–spread application of qualitative simulation techniques is the difficulty encountered when developing a qualitative model. Analyzing the resulting behavioral description and revising the model in response to this analysis requires a significant amount of expertise and is often left up to the modeler. As a result, developing a qualitative model is difficult for users who are not familiar with the field. Furthermore, this process is often not addressed within the literature making it difficult for such a user to obtain the necessary expertise except through trial and error. This paper addresses the process of model revision and presents a set of tools and methods to assist in the performance of this task. It also demonstrates how qualitative simulation can be used to obtain a more detailed understanding of the dynamical properties of the modeled system. The tools presented help the modeler extract information from a complex behavioral description by providing alternative views of the description, allowing the modeler to perform a focused search of the potential state space and providing explanation facilities for the branches occuring within the description. These tools and the methods are discussed with respect to the development of a semi–quantitative model of a controller for a tank.

Introduction

Qualitative simulation (Forbus 1984; Kuipers 1994; de Kleer and Brown 1984) is used to derive a behavioral description from a model of an imprecisely defined dynamical system to perform tasks such as monitoring, diagnosis, and design. It generates a symbolic description of all potential behaviors of the modeled system highlighting qualitative distinctions identified within the model. These qualitative distinctions may provide information that is not made explicit by a numerical simulation (*e.g.* quiescent states or classification of oscillatory behaviors as increasing, decreasing or harmonic). As with other simulation techniques, the benefits obtained are highly dependent upon the quality of the model provided as input. This is particularly true of qualitative models since an ambiguous model often results in an ambiguous behavioral description that includes a wide variety of potential behaviors.

The difficulty encountered developing a model is a major factor hindering the wide–spread application of qualitative simulation techniques. In general, model building research has focused on automated modeling techniques (Nayak 1994; Rickel and Porter 1997) and higher level modeling languages (Falkenhainer and Forbus 1991; Franke and Dvorak 1989). Frequently, however, models are developed by hand using an iterative process in which the modeler repeatedly revises the model in response to the results generated by the simulation. The process of analyzing the results of a simulation and revising the model is complex and requires a fair amount of expertise. As a result, it is difficult for users not familiar with the field of qualitative reasoning to quickly come up to speed and develop a model. Furthermore, this process is often not addressed within the literature (Weld and de Kleer 1990) making it difficult for such a user to obtain the necessary expertise except through trial and error.

In this paper we address both the *process* of revising a model and present a set of *tools* to assist a modeler in this process. The process focuses on the steps taken when revising a model and techniques for eliminating ambiguity. This process can also be helpful to relate features of the model with specific distinctions in the behavioral description. Model revision is a complex task that does not lend itself to a single, algorithmic method. The techniques presented here provide a road map to assist a modeler perform this task. The tools presented are used to filter the information provided by the simulation, selectively explore regions of the state space, and explain why certain phenomena did or did not occur.

The ideas presented here have been explored using the QSIM qualitative simulation algorithm (Kuipers 1994) although many of them can be generalized to other qualitative simulation algorithms. All tools have been implemented and will be available in the upcoming revision of QSIM soon to be released.

Qualitative simulation and model revision

The QSIM algorithm uses an abstraction of a ordinary differential equation (ODE) called a qualitative differential equation (QDE) to specify structural constraints between related variables within the modeled system. A tree of alternating time–point and time–interval states is generated describing all potential behaviors of the system consistent with the model following from a partially specified initial state. Each path within the tree corresponds to a qualitatively distinct behavior. Each state provides a qualitative magnitude (qmag) and direction of change (qdir) for each variable. The qdir is either increasing (inc), steady (std) or decreasing (dec) while the magnitude is defined on a totally ordered set of landmark values. An *event* occurs whenever a variable reaches a landmark or becomes steady. Landmark values are introduced to represent critical points identified during the simulation. Quantitative information can be added to the model in the form of numeric bounds and static functional envelopes (Berleant and Kuipers 1988; Kay and Kuipers 1993).

Branches occur within the behavioral description whenever the model provides insufficient information to disambiguate between alternative successor states. In general, an ambiguous, loosely constrained model results in a complex behavioral description with a large number of behaviors. Often, additional information not included within the model is available and the model must be revised to incorporate this information.

Model Revision

Model revision is the process of analyzing the results of a simulation, comparing these results against the modeler's expectations and revising the model as required. When a mismatch occurs between the modeler's expectations and the results of the simulation, the modeler must determine whether it is the model or his expectations that are incorrect. If the model is too detailed, incorrect or incomplete, the model is revised and re–simulated and then the process repeats itself.

The following questions are addressed at various points throughout this process:

- **What is described by the behavioral description?** To evaluate the accuracy of a model, the modeler must have an understanding of the classes of behaviors described by the resulting behavioral description.
- **Why do certain phenomenon occur within the description?** Often specific events or branches within the behavioral description must be understood with respect to the structure of the dynamical system for the modeler to appropriately evaluate the results of the simulation.
- **How can the model be revised if required?** Finally, the modeler must compare the results against his expectations and determine if the model needs to be revised. If revision is required, the modeler must determine how to revise the model to account for the observed mismatch.

Addressing these questions is a fairly straight–forward task if a simulation yields a small number of behaviors. For complex behavioral descriptions, however, it is often quite difficult to address these questions due to the variety of interactions that occur between different variables within the model. Furthermore, the initial version of a model is almost invariably under-constrained due to omitted information. Thus, even for tightly constrained systems that will eventually result in a small number of behaviors, the modeler must still deal with a complex behavioral description during the model revision process.

In the next section, we present a set of tools designed to assist a modeler answer these questions. This is followed by a discussion of the process of model revision in which each of these questions is addressed and a collection of model revision techniques presented that make use of these tools. A sequence of figures are used to provide a case study demonstrating the application of both the techniques and the tools in the development of a non–trivial example of a controlled tank. The steps described correspond to actual steps that were taken during the development of this model in collaboration with Bjarne Foss, a control theorist at the Norwegian University of Science and Technology in Trondheim. The model was developed to explore how qualitative simulation could be used to perform robustness analysis for a class of controllers.

Behavior Analysis Tools

This section describes a set of tools to analyze and understand the results of a qualitative simulation. How each of these tools can be used during the model revision process along with examples demonstrating the application of these tools is provided in section .

Temporal logic trajectory constraints

TeQSIM (Brajnik and Clancy 1996) is an extension of the QSIM algorithm that incorporates temporal logic model checking into the qualitative simulation process.

Propositional formulæ specifying qualitative and quantitative state information are combined using temporal operators such as `until` and `next` to form path formulæ that specify trajectory information. The simulation is restricted to behaviors that can potentially model these temporal logic expressions. TeQSIM can be used to restrict the region of the state space that is explored during a simulation thus simplifying the complexity of the behavioral description. TeQSIM can be also used to control the behavior of the exogenous variables that drive the system, it allows the modeler to specify discontinuous changes in exogenous variables and to inject temporally bounded external events into the simulations.

Trajectory constraints can also be applied following completion of a simulation to filter the resulting behavioral description. This allows the modeler to interactively explore different regions of the state space contained within the behavioral description.

Finally, temporal logic expressions can also be used to query the results of a simulation to determine if the resulting behavioral description models the temporal logic expressions (Kuipers and Shults 1994a; 1994b). This allows the modeler to evaluate whether certain desirable or undesirable properties are exhibited by the modeled system when evaluating the results.

Selective Continuation

Selective continuation allows the modeler to gradually extend individual, incomplete behaviors without extending the entire tree. This is useful when a model results in very large or infinite behavioral description to manually control the region of the state space that is explored. The agenda that is used to control the states that are simulated is modified to contain only un–simulated states within the behaviors identified by the modeler. The modeler also specifies the extent to which these behaviors are extended.

Variable Focus

Variable focus generates a more abstract representation of the behavioral description called a *view tree* that is a projection of the description onto a subset of the variables within the model. A branch only occurs within a view tree when a distinction occurs in one of the *viewed* variables. These are the only variables that are described by the states within a view tree. A mapping is maintained between states within a view tree and the main behavioral description. View trees are incrementally generated during the simulation.[1] Figure 1 describes the algorithm used to generate the view tree.

A view tree satisfies all of the properties of a standard behavior tree. Landmarks are created when appropriate and quiescent, cycle, and transition states are all represented. A mapping is maintained between the landmarks in the view tree and the landmarks in the main behavior tree. In addition, the user is able to interact with the view tree using the same user interface developed for exploring normal behavior trees. In particular, temporal logic expressions can be used to query or filter the view tree and selective continuation can be used to extend the simulation for incomplete behaviors by mapping these behaviors to their corresponding behaviors within the main tree (see figure 3).

Variable focus has been available as part of the QSIM distribution for a couple of years although it has recently been extended to include some of the functionality described above. It is similar in concept to a technique recently developed by Mallory (Mallory, Porter, and Kuipers 1996) to abstract envisionment graphs. Variable focus, however, is designed to work with a behavior tree and thus must handle the introduction of landmarks.

Event Analysis

Event analysis assists a modeler in determining why a branch occurs within the behavioral description by identifying the relationships between the events that cause the branch to occur. Identifying the events that occur following a given state and understanding how they are related can be quite complicated especially when there are a large number of potential successor states. An event occurs when a variable crosses a landmark or becomes steady. Branch points within the behavioral description result whenever there are multiple events that can follow a given state and the model fails to completely constrain the ordering of these events. Many events, however, often occur in unison following a specified state due the relationships between variables within the model. A *compound event* is a set of events, such that if one event within the set occurs following a given state, then all of the events occur. For example, if two variables are related by a monotonic function constraint and one of the variables becomes steady, then the other variable must also become steady.

Event analysis analyzes a branch point as follows:

1. All compound events are identified.

2. A causal analysis is used to analyze the relationships between the variables within a compound event and identify a *primary event* or a set of primary events.

[1] For clarity, several details concerning transition states and handling of successor relationship have been omitted.

Conditions	Actions
• S_M is an initial state	• Create a new view state.
• $S_M =_v$ predecessor(S_M) • if S_M is a time–interval state then Π(predecessor(S_M)) must also be a time–interval state.	• $\Pi(S_M) := \Pi$(predecessor(S_M))
• $S_M \neq_v$ predecessor(S_M) • $\exists S' ::\ S' \in T_V \wedge S' \in$ successors(Π(predecessor(S_M))) $\wedge$ $S' =_v S_M$	• $\Pi(S_M) := S'$
• none of the preceding conditions are true	• Create a new view state S' such that $S' =_v S_M$ • Add S' to successors($\Pi(S_M)$).

- As each new state S_M is created within the main tree, the rules in the preceding table are used to determine whether the new state should be mapped to an existing state within the view tree T_V or if a new view state should be created.
- The relation $=_v$ evaluates to true when two states are equivalent with respect to the set of viewed variables v and their statuses are compatible (*e.g.* a final state cannot be equivalent to a non–final state). The symbol $:=$ is an assignment operator.
- The mapping function $\Pi(S)$ maps state S to the corresponding state within the view tree.

Figure 1: Variable focus state processing algorithm

3. A logical representation is used to describe the relationships between separate compound events.

Compound events are identified through an analysis of the successors states for a given state S. All of the events that occur in at least one successor of S are identified. These events are then partitioned such that two events are contained within the same partition if and only if they always occur together in the successors of S. A compound event is created for each partition. Compound events are labeled using alphabetic characters that are used when identifying dependencies between compound events.

A heuristic that is based upon a causal analysis (Iwasaki 1988; Nayak 1994) of the model is then used to identify a primary event or set of events that is highlighted when the compound event is displayed to the user (see figure 6).

Identifying the relationships between compound events is the most useful feature of event analysis. Each successor state is labeled using the compound events that occur within the state. For example, if there are two compound events A and B and their ordering is unrelated, then there will be a total of three successor states, S_A, S_B and S_{AB}, where the subscript represents the state's label. The set of labels for the successor states can be viewed as a propositional logic expression. For the example above, the expression would be:

$$((A \wedge \neg B) \vee (\neg A \wedge B) \vee (A \wedge B))$$

This expression is in disjunctive normal form and can be translated into a slight variation of implicative normal form through a straight–forward transformation algorithm that results in the following expression

$$((A \vee B))$$

meaning that the occurrence of compound event A and compound event B are unrelated.

Implicative normal form is similar to conjunctive normal form except negated literals within each clause are eliminated by translating the clause into the form

$$(\texttt{IF}\ (a_1 \vee a_2 \vee \ldots a_m)\ \texttt{THEN}\ (c_1 \vee c_2 \vee \ldots c_n))$$

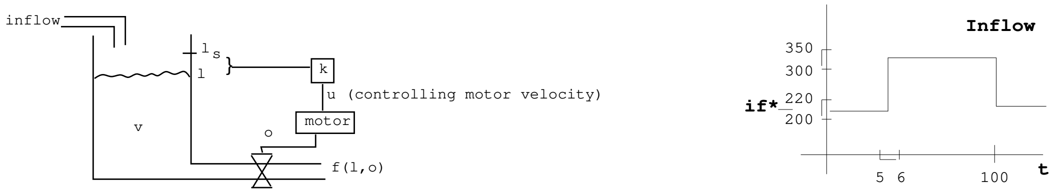
where $a_1 \ldots a_m$ correspond to the negated literals and $(c_1 \ldots c_n)$ correspond to the positive literals. A number of simplification rules are then applied to combine similar expressions so that the information is presented to the modeler in a concise and meaningful manner.

For example, suppose there are a total of four compound events, A through D, that result in three states that are labeled as follows: S_{AB}, S_{AC}, S_{ABCD}. Following conversion, this results in the following set of clauses and interpretations:

Clause	Meaning
(A)	A always occurs
$(B \vee C)$	B and C are independent
$((B \wedge C)\ \texttt{IFF}\ (D))$	B and C occur together if and only if D occurs.

In addition to the dependencies between compound events, each successor state is displayed with its label so that the user can evaluate how they combine independent of the analysis provided above.

By representing the dependencies in this form, it is easy to determine the correlations between the compound events within the successor states. These correlations can be related back to the constraints within



(a) Controlled tank

(b) Inflow wave perturbation

Problem: Develop a proportional controller to control the level in a single–input, single–output tank by operating on the *velocity* of the valve (a). Reason about the closed loop behavior of the system under various perturbations to the input and characterize the potential trajectory space of the system given a certain degree of uncertainty.

Analytic Model:

$$\begin{cases} \dot{v} &= \mathit{inflow}(t) - f(l,o) \\ l &= g(v) \\ o &= h(u) \\ \dot{u} &= k(l - l_s) \end{cases}$$

where v, l are volume and level of liquid in the container, $o \in [0,1]$ is the opening of the valve, $f(\cdot,\cdot)$ gives the outflow, u is the control signal, l_s is the set point, and k is the controller's gain. It is assumed that k and l_s are constant, $f \in M^{++}$, $g \in M_0^+$ and $h \in M_0^+$ and saturates at 0 and at 1. (M_0^+ denotes the class of monotonically increasing functions that pass through the origin, while M^{++} the class of two–valued functions that are monotonically increasing in both variables.)

Quantitative Information: Quantitative bounds are provided for the landmarks and monotonic envelopes for the functions accounting for the range of uncertainty.

Controlling the inflow: Trajectory constraints are used to control the behavior of the inflow. For the scenario discussed here a wave perturbation (b) is specified in which the inflow increases and then decreases. The following trajectory constraints define two temporally bounded external events (`step-up` and `step-down`), and specify discontinuous changes in the inflow corresponding to these events along with a bound on the new value following the initial perturbation.

```
(event step-up :time (5 6))
(event step-down :time 100)
(disc-change (event step-up) ((inflow (0 inf)) :range (300 350)))
(disc-change (event step-down) ((inflow if*)))
```

Figure 2: Control example description

the model to gain an understanding of why the branch occurred.

Successor Explanation

Successor explanation answers the question:

Why can't event E occur following state S?

where E provides a partial state description called a *template*. For example, the modeler might want to know why a variable that is increasing does not becomes steady.

Successor explanation attempts to generate a successor state for S that matches the template by inferring qualitative value information for each variable. Inferences are made by propagating the qualitative values in the template, applying continuity constraints to S, and identifying variables that are constrained from changing by the model. Support for each inference is recorded in a directed dependency graph. When an inconsistency is detected, the dependency graph is used to generate a proof tree explaining why a successor state matching the template does not occur. The explanation presented to the user is categorized into one of five different explanation types. Table 1 describes these categories while table 2 describes the algorithm that is used to construct an explanation. Figure 6 provides an example demonstrating how successor explanation can be used while figure 7 presents the proof tree for this example.

Revising a Model

Model revision is a complex, unstructured process that currently requires a significant amount of expertise and experience developing qualitative models. In addition, it requires at least a partial understanding of the dynamics of the modeled system so that the modeler can evaluate the results of the simulation. This process has

Exp-1	**Successor exists:**	A successor matching the template exists.
Exp-2	**Inconsistent template:**	The template provided by the modeler is inconsistent with respect to the model.
Exp-3	**Continuity violated:**	A consistent state matching the template cannot be reached due to continuity.
Exp-4	**Consistent state filtered:**	After the consistent successors of a state are computed, QSIM uses a set of auxiliary state filters to apply additional constraints. (*e.g.* the energy constraint, higher–order derivative information, quantitative information, etc.) If a state matching the template is consistent with the constraints in the model but is filtered by a state filter, information about the filter is provided.
Exp-5	**Constraints violated:**	Constraint satisfaction prevents a consistent state matching the template from being constructed.

Table 1: Explanation types

Step 1:	If a successor state matching the template exists return *exp-1*. (see table 1)
Step 2:	Propagate the values in the template through the constraint graph. Record information in the dependency graph as each propagation is made. If propagation results in an inconsistent set of values, return *exp-2*.
Step 3:	Assert qualitative value information required by constraints in the model given the current state. For example, if a variable is restricted by a `constant` constraint, then its qualitative value can be asserted.
Step 4:	Use continuity to infer an abstract qualitative value for each variable. This value is the union of the possible successor values for each variable generated by QSIM. The constraints are evaluated to determine if this information is consistent with the current information within the partial state. If an inconsistency occurs, return *exp-3*.
Step 5:	If an explanation has not been identified, the QSIM successor generation function is called and information is recorded as it identifies the valid successor states. This information is then presented to the user in the form of *exp-4* or *exp-5*. In most instances, the inferences made in the first four steps are adequate. A proof tree is not provided for these last two explanation types.

Table 2: Successor explanation algorithm

been broken down into three phases that are characterized by the *what*, *why* and *how* questions presented earlier. This section presents techniques for addressing each one of these questions. While we do not suggest that this set of techniques is comprehensive, they do provide a road map that helps to structure the inherently unstructured task of model revision. A great deal of overlap exists between these phases and thus there is a significant amount of interaction between the steps taken within each phase.

Figure 2 describes an example model that is presented as a case study demonstrating many of the techniques and tools presented here.

What happens?

For complex behavioral descriptions, it is often difficult to gain a global perspective of the classes of behaviors described. This is even more difficult when the system exhibits oscillatory behavior that results in an infinite simulation. The following techniques have all been used effectively for both model revision and to answer the question of what happens within a behavioral description. Often, multiple techniques are employed on the same model to understand different phenomenon and different regions of the trajectory space.

1. **Focus on individual variables.** Variable focus allows the modeler to restrict the behavioral description to a subset of variables. Viewing variables independently or in small subsets greatly simplifies the task of understanding the region of the trajectory space described by the behavioral description. Often this region may match the modelers intuition, but it is hard to make this evaluation without focusing on individual variables (see figure 3).

2. **Independently explore subregions of the trajectory space.** Potential interactions between a large number of phenomenon often makes it difficult to understand what occurs under what situations. Trajectory constraints can be used to explore subregions of the trajectory space independently (see figure 5).

3. **Gradually extend individual behaviors or sets of behaviors.** Selective continuation allows the modeler to select individual behaviors to be extended. Often, a large number of behaviors exhibit similar behavior. For complex behavioral descriptions, it is often best to iteratively select individual behaviors or groups of behaviors for simulation. An analysis of these behaviors resulting in an un-

derstanding of the distinctions that occur can often be generalized to the rest of the behavior tree (see figure 3).

4. **Analyze individual branches.** The global behavior of the system emerges due to local distinctions that occur during the simulation. Understanding these local distinctions often leads to a more general understanding of the classes of behaviors described. Event analysis can be used to analyze individual branches within the tree to gain an understanding of the distinctions that are occuring. Often these distinctions repeatedly occur throughout the simulation and patterns can be observed in the behavioral description. Observing these patterns is particularly helpful when coupled with the selective extension of the behavior tree. It is often best to start with either the initial branches within the tree or the most complex branches that occur (see figure 5).

Why does it happen?

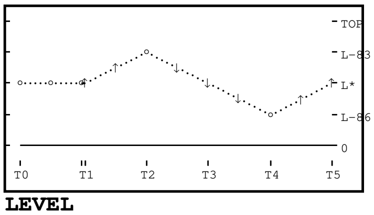
Understanding why certain phenomenon do or do not occur requires a mapping between certain properties of the model and the behavioral distinctions that result from these features. This mapping is used to evaluate the appropriateness of the model and to hypothesize and evaluate potential revisions to the model. For a complex behavioral description, it is often best to obtain such an understanding by analyzing individual points within the description. For any given state S within the behavioral description, two questions can be asked with respect to the successors of S:

- Why *does* a branch occur following S?, and
- Why *doesn't* a branch occur?

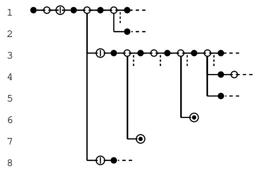
The event analysis tool is helpful in addressing the first question by providing a detailed analysis of the dependencies that exist between the events that lead to a particular branch (see figures 5 and 6). This analysis can then be used by the modeler to map these relationships back to the constraints within the model.

The second question is addressed using the successor explanation tool which describes the relationships between the constraints and the current state that prevent the partial state from occuring (see figures 6 and 7).

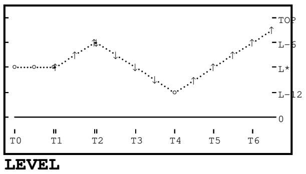
These two tools provide a local understanding of the events that occur within the description. Often, a more global understanding is obtained by revising the model and comparing the results of the two simulations to determine how the revision affected the behavioral description. Model revision is discussed in detail in the next subsection.



(a) Behavior 9 from initial simulation



(b) Selective continue `Level` view tree



(c) Behavior 3 from (c)

The modeler expects to observe two equilibrium points corresponding to the two distinct values for the input variable inflow. These equilibrium points are each approached by an oscillatory behavior due to the action of the controller. To test the model, we attempt to determine if a non–decreasing oscillation occurs. If one can be identified, then the model is incorrect and it must be revised.

- Simulation of the initial model with a state limit of 100 generates a complex behavior tree with 56 behaviors and takes a total of 1 minute and 48 seconds on a Sparc 10 to generate. Due to the complexity of the description it is hard to understand what is happening. If we use variable focus to describe the behavior of `level`, there are a total of 8 behaviors but none of them are extended sufficiently to determine if decreasing oscillations occur (a)..
- Alternately, we could simulate the model with a state limit of 20, use variable focus to view the results, and then use selective continuation to extend individual behaviors. After two extensions, a view tree for `level` results in 8 behaviors (b). One of these behaviors demonstrates that a non–decreasing oscillation can occur (c). This method only requires a total of 24 seconds simulation time. (The dotted lines in the view tree (b) correspond to states that map to states within the main tree that still must be simulated.)
- By analyzing the model, it can be determined that explicit energy terms must be represented within the model to appropriately model the system.

Figure 3: Initial controlled tank simulation

- After revising the model to incorporate an energy constraint, iterative simplification is used to simplify the model to its basic structure. Not only does this assist in developing the model, but it also increases the modeler's understanding of the relationships between the structural properties of the model and the corresponding behavioral features.

  The initial equation following addition of the energy constraint contains a conservative and a non–conservative term resulting in a complex model that requires the introduction of a number of new variables. (In this model, $\ddot{u} = k\frac{\dot{v}}{A}$, it is assumed that $f(l,o) = f_1(l) \cdot f_2(o)$, $\xi = f_1(l_s)$ and $h_1(x) = f_1(\frac{x}{k} + l_s) - \xi$.)

  $$\ddot{u} = \left[\frac{k}{A}(\mathit{inflow} - \xi \cdot f_2(h(u)))\right] + \left[-\frac{k}{A}h_1(\dot{u}) \cdot f_2(h(u))\right] \quad \text{(Rev-1)}$$

  Abstracting away constants results in the equation:

  $$\ddot{u} = \mathit{inflow} - f_2(h(u)) - h_1(\dot{u}) \cdot f_2(h(u)) \quad \text{(Rev-2)}$$

  Removing the monotonic functions f_2 and h_1 results in the equation:

  $$\ddot{u} = \mathit{inflow} - h(u) - \dot{u} \cdot h(u) \quad \text{(Rev-3)}$$

  Finally, we remove the saturation function h which results in the basic model

  $$\ddot{u} = \mathit{inflow} - u - \dot{u} \cdot u \quad \text{(Rev-4)}$$

  While a direct mapping does not exist between the variables within this model and the physical component, for the sake of understanding the simulation results it may be helpful to think of $\dot{u}$ as the level of the tank and $\ddot{u}$ as the netflow.

- A qualitative model is developed for `Rev-4`. Once the model is understood, we add back the information needed to regenerate `Rev-3` and so on until we reach `Rev-1` (see figure 6).

Figure 4: Applying iterative simplification to the controller model

How can the model be revised?

Determining how a model can be revised is difficult since so many different options can be considered. For an over–constrained model, this process is not too difficult since the search for potential revisions is limited to the information contained within the model. Furthermore, the successor explanation tool can be used to isolate the source of the problem. Unfortunately, it is more common for the model to be under–constrained thus resulting in an overly general behavioral description. Such a problem occurs if either the model does not include information that is available to the modeler or if the model is too detailed in its description of the dynamical system. It is also possible that the generality of the description is a reflection of the actual complexity of the dynamics entailed by the model. Even in this situation, however, to assist in the analysis of the results the modeler might identify additional assumptions that can be incorporated into the model to produce a more restricted set of behaviors.

Insufficient information It is quite common for a modeler to omit information that is important in restricting the behavior of the system during simulation. These omissions occur because qualitative model descriptions are weaker than numerical ones, and so information implicit in a precise numerical model may not appear when this model is directly translated into qualitative terms. The following classes of information are often the most beneficial when attempting to restrict an ambiguous description:

1. **Energy constraints** — An energy constraint explicitly describes the change of energy within a system in terms of its conservative and non–conservative components. Thus, behaviors in which the sum of the conservative and non–conservative components do not add up to the net energy change can be refuted. Clearly, energy constraints should be included in models that have this property or a derivable Lyapunov function. While energy constraints will increase the number of variables in a model, their constraining power is usually worth the cost. See (Fouché and Kuipers 1992) for a detailed discussion.

2. **Corresponding values** — Corresponding values help to pin down the curve defined by a constraint by introducing qualitative points through which it must pass. For example, given the constraint that $A + B = C$ and knowledge that the (existing) landmarks a, b, and c satisfy the equation, the corresponding value (a, b, c) should be included in the

constraint definition. Thus, the simulator can refute states which are inconsistent with this constraint. Note that corresponding values should only be added between existing landmarks. Adding new landmarks to the quantity spaces of variables provides no additional constraining power.

3. **Redundant constraints** — In some models, it is possible to state a constraint in more than one way. While this is clearly redundant analytically, it is possible that ambiguity in one formulation may not be present in the other. Kuipers (Kuipers 1994) provides an example of redundant constraints in modeling a hot/cold mixing tank.

4. **Quantitative information** — In some cases, the qualitative behavior of a model may be inherently ambiguous. In cases where simplification is an unacceptable alternative, quantitative information may be available that can reduce ambiguity (Kay 1996).

Event analysis is often helpful in focusing the search for what information has been omitted by analyzing unexpected branches. A useful strategy is to restrict the behavioral description to either the desired behaviors or the undesired behaviors and to analyze these sets independently to try and observe relationships between variables that may not be specified within the model (see figure 3).

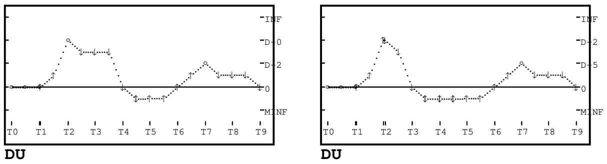
A modeler should be wary of attempting to restrict an ambiguous behavioral description by adding variables to the model. While providing additional constraints and/or corresponding values does not increase the complexity of the description, adding variables often leads to additional branches. While increasing the overall complexity of the simulation, new variables will not increase the complexity of the description from the perspective of the prexisting variables. Thus, by using variable focus to ignore these added variables, the additional complexity of the description can be eliminated.

Overly–detailed models Qualitative simulation describes all potential behaviors of the system that are consistent with the qualitative description provided. This often includes unlikely behaviors that cannot be eliminated due to the inherent limitations of the qualitative algebra when a variable is dependent on two opposing influences. For example, when a bathtub is filling with water, the effect of evaporation is often considered insignificant with respect to the netflow of water. If a qualitative model included evaporation, however, the behavioral description would include a behavior in which evaporation becomes the dominate influence and the level in the tank decreases. This is an example of an overly–detailed model resulting in

```
Label Primary Events          Dependent Events
----- --------------          -------------------
  A:  Ext-event: step-down
  B:        DDU: 0            DU: std

Event Dependencies:           (A v B)
State-labels:                 S-203: (A)
                              S-204: (B)
                              S-205: (A ^ B)
```

(a) Event analysis for first branch



(b) Before down step

(c) After down step

- Simulation of the simplified model (`Rev-4`) with a state limit of 60 still results in an infinite, fairly complex behavioral description that is hard to understand. To understand the structure of the tree we will begin by analyzing individual branch points.

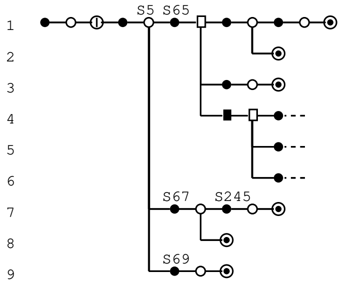
- The results of event analysis following the first branch point identifies two unrelated, compound events: `DU` becomes steady or the step decrease in `inflow` occurs (a). Further analysis of the middle branch shows a similar branch occurs repeatedly, *i.e.* `DU` can oscillate an arbitrary number of times before the `step-down` occurs.

- Trajectory constraints can be used to divide the behavioral description of the system into the region before `step-down` occurs and the region after `step-down` occurs.

  1. The trajectory constraint `(never (event step-down))` prevents `step-down` from occurring resulting in a simulation that is equivalent to a single step increase. Variable focus generates a view tree with a single behavior (b). `dU` exhibits a decreasing oscillation.
  2. The trajectory constraint `(before (event step-down) (qvalue du ((0 inf) std)))` requires `step-down` to occur before `DU` reaches steady for the first time. Once again this results in a simple behavioral description describing a decreasing oscillation (c).

- Once a better understanding of the behavior is obtained, the simulation is run without the additional trajectory constraints and the resulting description queried using temporal logic expressions to confirm that it exhibits a pair of decreasing oscillations. While a complex behavioral description still results, our analysis has provided us with an understanding of the description and confidence that the model is correct.

Figure 5: Simulation of simplest model (`Rev-4`)

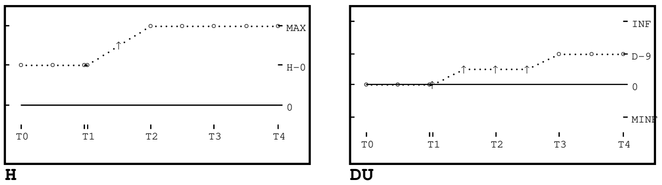


(a) Partial behavior tree

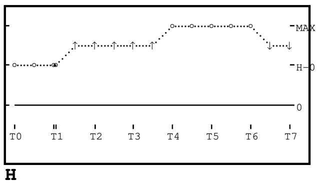
```
Label Primary Events     Dependent Events
----- --------------     --------------------
  A:  DDU:   0           DU: std
  B:    U:   1            C: std H:(max std)
  C:    Z:  std

Event Dependencies:      (C) IFF (A ^ B)
                         (A v B)
State-labels:            S-65: (A)
                         S-67: (B)
                         S-69: (A ^ B ^ C)
```

(b) Event analaysis output for state S-5



(c) Behavior 7 from partial tree



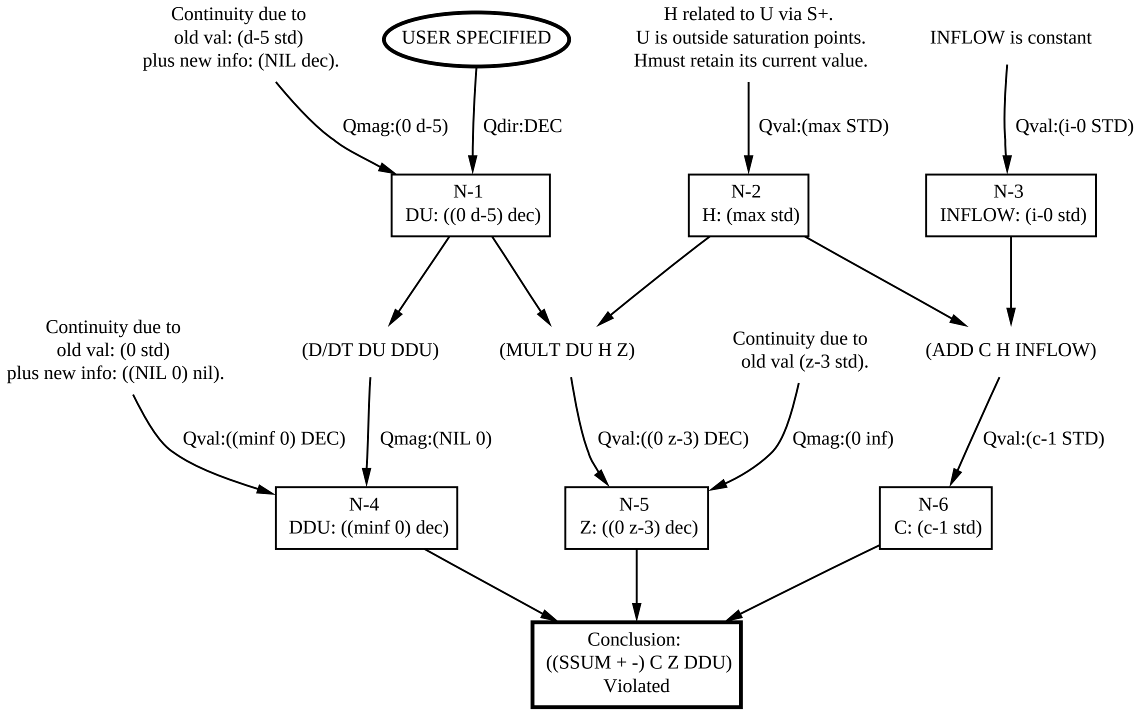
(d) Extension of behavior 6

After understanding the behavior of the basic model (`Rev-4`), the saturation function h is incorporated back into the model to create a qualitative model corresponding to `Rev-3`. Analyzing the simulation results for this model explains how the saturation function affects the behavior of the system and strengthens our confidence in the accuracy of the model.

- A new branch occurs in the behavioral description at state `S-5` following the initial perturbation (a). Event analysis reveals that the branch results from two unrelated events: the control signal `U` reaching the saturation landmark 1 and the derivative of `U` (*i.e.* `DU`) becoming steady (b). (`Z` is an intermediate variable within the model that equals $h \cdot \dot{u}$.) Note how the event analysis describes the dependencies between the compound events.
- For the bottom two branches in which the saturation landmark is reached, all of the behaviors result in the tank reaching an equilibrium point that is greater than the set point (c). Thus, the controller fails to regulate the tank level as desired in this situation. (Recall that `DU`=0 when $l = l_s$).
- As the rest of the tree following the first branch is extended, however, other behaviors are identified in which H reaches saturation and then eventually decreases (d).
- To understand the distinctions between these behaviors and confirm that the model is behaving appropriately, successor explanation is used to explain why the level in the tank, which corresponds to `DU`, cannot begin to decrease following time–point `t3` in behavior 7 (see figure 7).

Once an understanding of `Rev-3` was obtained, the process continued until a qualitative model matching `Rev-1` was derived. Simulation of this model resulted in a behavioral description that matched our intuitions. Iterative simplification not only helped in the development of the model, but it provided an understanding of the branches within the description with respect to the structural properties of the model.

Figure 6: Successor explanation applied to model `Rev-3`



- To gain an understanding of the system and confirm that the model is behavior correctly, successor explanation is used to answer the question

  Why can't `DU` begin to decrease after state `S-245`? (see figure 6.) This corresponds to the time interval $(t_3\ t_4)$ in figure 6(c).

- Successor explanation generates a proof tree explaining why a state matching this description is prevented. The boxed nodes within the proof tree correspond to instantiated variables and the arcs correspond to inferences. Root nodes are identified using text that describes the source of the information for an inference. (The figure above is the actual output generated using a graph layout program called `dot` obtained from Bell Labs.)

- The nodes in the proof tree are inferred as follows:

N1:	The user specified that the qdir of `DU` is `dec`. Since it was previously `(d-5 std)`, it must now be `((0 d-5) dec)`.
N2,N3:	Since `U` is above the saturation threshold, `H` must remain `std`. `Inflow` is constant so its value is also asserted.
N4:	Since `DU` is decreasing, `DDU` must be negative. Since it is currently `(0 std)`, continuity requires it to also be `dec`.
N5,N6:	`Z` must be positive and decreasing due to the `mult` constraint and `C` must be `(c-1 std)` due to the `add` constraint.
Conclusion:	The infered values for `DDU`, `Z` and `C` are inconsistent with the `ssum` constraint.

- For a state matching the template to be consistent, `DDU` must already be negative when the saturation threshold is crossed so that it can be increasing when `DU` begins to decrease. (*i.e.* If this occurs, the `SSUM` constraint will no longer be violated.) This observation is consistent with the behaviors in which saturation is reached, but the controller is still able to regulate the tank.

- Therefore, the level in the tank must be decreasing when the saturation point is reached if the controller is to regulate the tank appropriately.

Figure 7: Successor explanation proof tree

unexpected behaviors. In addition, an overly–detailed model may simply complicate the resulting behavioral description making it difficult to evaluate and isolate the effect of various features of the model.

We call the technique for refining such overly–detailed models *iterative simplification.* Iterative simplification is a two step procedure in which the model is reduced to a more basic representation so that the output can be easily analyzed. Then the original model is reconstructed by gradually reincorporating the additional information and analyzing the results of the simulation as each new bit of information is added.

At the heart of iterative simplification are the methods for model simplification:

1. **Algebraic simplification** is based on two very simple rules:
   - variables related by a monotonic function can be collapsed into a single variable within the model, and
   - multiplicative constants can be removed.

   Assuming that the state variables are left unchanged, the application of the rules results in a reduction in the complexity of the behavioral description due to the elimination of intermediate and output variables along with their landmarks. Figure 4 demonstrates how this technique has been applied.

2. **State simplification** reduces the number of state variables within a model. Since the size of the state space affects the complexity of a model, fewer state variables can simplify the behavior tree. One example of state simplification is the use of time–scale abstraction (Kuipers 1994) where state variables can be eliminated by viewing their processes as instantaneous with respect to other model processes. Another example is the combination of multiple, related, state variables into a single state. Kay (1992) applies state simplification to a model of the Space Shuttle Reaction Control System.

3. **Order–of–magnitude reasoning** introduces additional information into the model that can disambiguate the relative scale of differing influences within a model. Within the QSIM framework, the `W+` constraint can be used for this purpose (Kay 1992)).

4. **Operating region partitioning** splits the behavior of a single model into pieces, each of which is a simpler model. Since each model is simpler, it produces fewer behaviors. The cost, however, is that rules for translating between models must be described. Kay (1992) uses operating region partitioning when modeling a pressure regulator.

5. **Decoupling the model** isolates the behavior of subcomponents within the model, each of which can be examined separately. Decoupling relies on user–specified trajectory constraints (described using TeQSIM) to control the behavior of particular model variables, (such as flows, which typically link different components of the model). Providing a trajectory constraint effectively turns these endogenous variables into exogenous ones, thus reducing the resulting behavioral complexity.

   Another decoupling strategy uses trajectory constraints to completely model a subcomponent of a model. For example, one could decouple a plant from its controller by specifying the behavior of the controller completely via trajectory constraints and then evaluate the effect of different controllers on the behavior of the plant.

Discussion and Future Work

Model revision is a complex process that requires an understanding of and experience with both the problem domain and the modeling methodologies. The techniques presented here are an initial effort at simplifying this process and reducing the level of experience within the field of qualitative reasoning required to build a qualitative model. Due to the complexity of this process and the diversity of knowledge required, we feel that it is unlikely that the entire process can be effectively automated. Instead, we are more interested in developing techniques that help the modeler search the space of potential models thus simplifying the model revision process. Some potential extensions include:

- developing a more abstract modeling language that matches the representations used by engineers when reasoning about the behavior of a dynamical system,
- providing an algebraic manipulation and simplification component that can be used to automate the process of iterative refinement using algebraic simplification and identify redundant constraints that may further constrain the model,
- developing more sophisticated explanation techniques that can provide a comprehensive description of the classes of behaviors described within the behavioral description, and
- incorporating theory refinement techniques (Richards, Kraan and Kuipers 1992) to induce potential revisions to the model using the current model and a description of the expected, desired, or observed behaviors of the system.

Conclusions

Recently, a number of applications using qualitative reasoning techniques have begun to surface demonstrating the effectiveness of these techniques when addressing complex, real-world problems. In general, however, these applications do not use qualitative *simulation*. One reason for this is the difficulty encountered when developing a non–trivial, qualitative model due to the complexity of the model revision process. The complexity of this process often makes it difficult for researchers from other disciplines to apply qualitative simulation to tasks in which it may be suitable.

We have addressed the process of model revision from two perspectives. First, a set of tools have been presented that simplify the process of analyzing the results of a simulation to gain an understanding of the classes of behaviors exhibited by the model. Second, the process of model revision is discussed along with a presentation of a variety of techniques that can be used during this process.

Hopefully, this research coupled with recent developments that both reduce the complexity of a qualitative simulation (Clancy and Kuipers 1997; 1997) and incorporate additional quantiative information will facilitate the application of qualitative simulation techniques to a variety of real–world tasks.

Acknowledgements

We would like to thank proff. Bjarne Foss for his input in developing the controller model and Ben Kuipers for many enlighting discussions. This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Qualitative Reasoning Group is supported in part by NSF grants IRI-9504138 and CDA 9617327, by NASA grants NAG 2-994 and NAG 9-898, and by the Texas Advanced Research Program under grant no. 003658-242. Source code for the techniques presented is included in the most recent QSIM release available at **http://www.cs.utexas.edu/users/qr/**.

References

D. Berleant and B.J. Kuipers. Using incomplete quantitative knowledge in qualitative reasoning. In *Proc. of the Sixth National Conference on Artificial Intelligence*, pages 324–329, 1988.

G. Brajnik and D. J. Clancy. Temporal constraints on trajectories in qualitative simulation. In B. Clancey and D. Weld, editors, *Proc. of the Tenth National Conference on Artificial Intelligence*, pages 979–984. AAAI Press, Aug. 1996.

D. J. Clancy and B. Kuipers. Model Decomposition and Simulation: A component based qualitative simulation algorithm. In *Proceedings from the Fourteenth National Conference on Artificial Intelligence, AAAI-97*, Providence, RI, July 1997.

D. J. Clancy and B. Kuipers. Static and dynamic abstraction solves the problem of chatter in qualitative simulation In *Proceedings from the Fourteenth National Conference on Artificial Intelligence, AAAI-97*, Providence, RI, July 1997.

J. de Kleer and J.S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

B. Falkenhainer and K. Forbus. Compositional modeling: finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.

K. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

P. Fouché and B.J. Kuipers. Reasoning about energy in qualitative simulation. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(1):47–63, 1992.

D.W. Franke and D. Dvorak. Component–connection models. In *Model–Based Reasoning Workshop, IJCAI–89*, August 1989.

Y. Iwasaki. Causal ordering in a mixed structure. In *AAAI–88*, pages 313–318, 1988.

H. Kay and B.J. Kuipers. Numerical behavior envelopes for qualitative models. In *Proc. of the Eleventh National Conference on Artificial Intelligence*. AAAI Press/MIT Press, 1993.

H. Kay. A qualitative model of the space shuttle reaction control system. Technical Report AI92-188, University of Texas at Austin, Artificial Intelligence Laboratory, 1992.

H. Kay. SQsim: a simulator for imprecise ODE models. TR AI96-247, University of Texas Artificial Intelligence Laboratory, March 1996.

B.J. Kuipers and B. Shults. Reasoning in logic about continuous change. In *Principles of Knowledge Representation and Reasoning (KR–94)*. Morgan Kaufmann Publishers, Inc., 1994.

B.J. Kuipers and B. Shults. Reasoning in logic about continuous systems. In *8th International Workshop on Qualitative Reasoning about physical systems*, pages 164–175, Nara, Japan, 1994.

B.J. Kuipers. *Qualitative Reasoning: modeling and simulation with incomplete knowledge.* MIT Press, Cambridge, Massachusetts, 1994.

R.S. Mallory, B.W. Porter and B.J. Kuipers. Comprehending Complex Behavior Graphs through Abstraction. In *10th International Workshop on Qualitative Reasoning about physical systems*, pages 137–146, Fallen Leaf Lake, California, 1996.

P. Nayak. Causal approximations. *Artificial Intelligence*, 70:277–334, 1994.

B.L. Richards, I. Kraan, and B.J. Kuipers. Automatic Abduction of Qualitative Models In *Proc. of the Tenth National Conference on Artificial Intelligence.* AAAI Press/MIT Press, 1992.

J. Rickel and B. Porter. Automated modeling of complex systems to answering prediction questions. *Artificial Intelligence*, 1997. To appear.

D. Weld and J. de Kleer. *Readings in Qualitative Reasoning about Physical Systems.* morgan, Los Altos, CA, 1990.