Automatic Abduction of Qualitative Models

Bradley L. Richards
Dept. of Computer Sciences
University of Texas
Austin, Texas 78712
bradley@cs.utexas.edu

Ina Kraan
Dept. of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
Scotland, UK
inak@ai.ed.ac.uk

Benjamin J. Kuipers
Dept. of Computer Sciences
University of Texas, Austin
Austin, Texas 78712
kuipers@cs.utexas.edu

Abstract

We describe a method of automatically abducing qualitative models from descriptions of behaviors. We generate, from either quantitative or qualitative data, models in the form of qualitative differential equations suitable for use by QSIM. Constraints are generated and filtered both by comparison with the input behaviors and by dimensional analysis. If the user provides complete information on the input behaviors and the dimensions of the input variables, the resulting model is unique, maximally constrained, and guaranteed to reproduce the input behaviors. If the user provides incomplete information, our method will still generate a model which reproduces the input behaviors, but the model may no longer be unique. Incompleteness can take several forms: missing dimensions, values of variables, or entire variables.

1 Introduction

Qualitative simulation of physical systems provides researchers with insights by giving an overview of system behaviors without the deluge of detail inherent in quantitative simulation. Perhaps even more important, it may be possible to develop a qualitative simulation where developing a quantitative one would be impossible due to inexact knowledge of the system's internal workings. But even with the power of qualitative simulation systems like QSIM ([Kuipers, 1986, 1989]), developing qualitative models remains something of an art. For this reason, many researchers are investigating automatic model building. The most common approach is to construct models from given model fragments ([Forbus, 1984], [Forbus, 1986], [deKleer and Brown, 1984], and [Crawford, Farquhar, and Kuipers, 1990]).

In this paper, we present MISQ, a method for building models purely from behavioral information. Given some or all of the behaviors[1] exhibited by a particular system, we abduce a model which reproduces those behaviors. If the user provides sufficient information on the input behaviors, the resulting model is *unique*, *maximal*, and *correct* in the sense that it reproduces the input behaviors. The models we build are qualitative differential equations (QDEs) suitable for use by QSIM.

The QSIM framework provides explicit functions, landmarks, and corresponding values, all of which are critical to the success of MISQ. A weaker framework such as confluences would be inadequate.

MISQ was first implemented as a special-purpose system, and a preliminary description appeared in [Kraan, Richards, and Kuipers, 1991]. Since then, MISQ has been reimplemented as a domain theory within the general-purpose learning system Forte [Richards and Mooney, 1991]. The relational pathfinding method in Forte, described in [Richards and Mooney, 1992], allows MISQ to infer the existence of missing variables.

The remainder of this paper is organized as follows: Section 2 provides an overview of our method of model building. Section 3 proves the theorem central to the correctness of our approach. Section 4 gives detailed examples of models we constructed automatically. Section 5 presents our method for introducing new variables into a model. Section 6 summarizes related work. Finally, Section 7 gives our conclusions and suggests directions for further research.

2 Model Building in MISQ

Overview. MISQ's model generation process is broken into three major phases. In the first phase, if we are given quantitative data, we convert it into qualitative behaviors. It is also possible to input qualitative behaviors directly. In the second phase, we generate and test individual constraints, creating constraints consistent with the input behaviors. In the third phase, we construct models (QDEs) from the set of constraints generated in the second phase. If the models are not connected, i.e., they consist of independent sub-models, we use relational pathfinding to search for variables that connect the sub-models.

Conversion of quantitative data. We can execute on two forms of quantitative input: high-resolution sensor data or hand-generated quantitative behaviors. If the input is high-resolution sensor data, we convert the data to the required numeric precision and align events which occur in different variables at insignificantly different times[2]. We then discard all but the "interesting" points in the data; i.e., points where some variable reaches a maximum, a minimum, or zero. Hand-generated quantitative behaviors are the analog of processed sensor data: quantitative behaviors which include only interesting time points.

The quantitative behaviors are converted into qualitative behaviors. For each variable at each time point, the quantitative value is turned into a qualitative value consisting of a qualitative magnitude and a direction-of-change. The qualitative magnitude is constructed by generating a landmark value and, if it is new, inserting it into the qspace constructed so far[3]. The direction-of-change is determined by comparing the numeric value of the variable at the current time point with those of the preceding and subsequent time points. Further, we add qualitative states to behaviors as needed. If, for example, a variable is at a minimum at one time point and at a maximum at the next, the qualitative state for the interval during which the variable is increasing is added to the behavior.

Constraint generation. The input to the second phase is a set of consistent qualitative behaviors, the landmark values (qspaces) of the initial state, and dimensional information. In the first step of this phase, we select an arbitrary behavior and generate all constraints satisfied by any combination of variables. This is done by generating tuples of variables and testing their values against the satisfaction conditions for each constraint type. We do not generate tuples that lead to immediately redundant constraints, e.g., (M^+ x y) and (M^+y x).

We currently implement the following constraint types[4]: arithmetic constraints (add, mult, and minus), differential constraints (d/dt), functional constraints (M^+ and M^- for strictly monotonically increasing and decreasing functions), and direction-of-change constraints (constant). These constraints are a subset of the constraints provided in QSIM. Nevertheless, they are expressive enough to build many interesting qualitative models.

The satisfaction conditions are similar to those in QSIM, though somewhat simpler. Since the input behaviors are assumed to be correct, we need not check the continuity criteria from a state to its successor. And, since the satisfaction criteria within a state are the same for time points and intervals, we need not distinguish between time points and intervals. The constraint satisfaction criteria are based on the magnitudes, signs, directions of change, and corresponding values of the variables; these criteria are defined in detail in [Kuipers, 1986]. For example, the constraint (M^+ x y) is satisfied if the directions of change of x and y (expressed as increasing, decreasing, or steady) are always identical, and there are no conflicting corresponding values. If, for instance, there are corresponding values at (x_1 y_1) and (x_1 y_2), and y_1 and y_2 are known to be distinct values, the relationship between x and y cannot possibly represent a function. The constraint (d/dt x y) is satisfied if the direction-of-change of x is increasing, decreasing, or steady and the sign of y is +, -, or 0, respectively.

Finally, we ensure that the dimensions of the variables in each constraint are compatible. If, for example, the constraint (d/dt x y) has been generated, MISQ will test whether the dimensions of x can be the dimensions of y divided by time. This ensures that the constraints are abstractions of equations potentially representing real physical systems. Functional constraints impose no a priori restrictions on the dimensions of their arguments. Since we are working with qualitative data, dimensions are generally stated in terms of fundamental types like mass, time, or length rather than in units of measurement such as meters or grams. Since MISQ is only interested in the *relationship* between the dimensions of variables, users are free to define their own types, except for *time*.

In some cases, the user may be able to reduce the number of constraints in the final QDE by making dimensions more specific. For example, if a system contains both oxygen and water, and the user knows that it makes no sense to combine amounts of oxygen and water, the user can use different dimensions, such as amount-of-oxygen and amount-of-water.

After generating all possible constraints from a single behavior, we test all remaining behaviors against these constraints, eliminating any constraint that violates any satisfaction condition.

Model generation. If the user provided complete information on the input behaviors, the set of constraints from the second phase forms a unique model guaranteed to reproduce the input behaviors (see Section 3). If, on the other hand, the user provides incomplete information, the set of constraints from the second phase may not form a valid model. In this case, further processing is required (see Section 4).

Variable	Initial Qspace	Dimensions
Inflow	0, in1, ∞	mass/time
Outflow	0, ∞	mass/time
Netflow	-∞, 0, net1, ∞	mass/time
Amount	0, ∞	mass

State 0 (initial state)		successors: state 1
Variable	Magnitude	Direction-of-Change
Inflow	in1	steady
Outflow	0	increasing
Netflow	net1	decreasing
Amount	0	increasing

State 1		successors: state 2
Variable	Magnitude	Direction-of-Change
Inflow	in1	steady
Outflow	(0, ∞)	increasing
Netflow	(0, net1)	decreasing
Amount	(0, ∞)	increasing

State 2		successors: none
Variable	Magnitude	Direction-of-Change
Inflow	in1	steady
Outflow	out1	steady
Netflow	0	steady
Amount	amount1	steady

Figure 1. Single qualitative behavior of the simple bathtub. This information can be automatically generated from high-resolution sensor data.

As an example of model generation with complete information, consider an empty bathtub with a finite capacity, a constant inflow, and a constant drain opening. This bathtub exhibits three behaviors: reaching equilibrium at a level below the top of the tub, reaching equilibrium exactly at the top, and overflowing. We simulated the bathtub using QSIM, and presented MISQ with a complete qualitative description of the behavior with an equilibrium point less than full (see Figure 1). MISQ abduced the exact QDE used to produce the behavior, with the addition of two redundant constraints:

(constant inflow)
(add outflow netflow inflow)
(M- outflow netflow)
(M+ outflow amount)
(M- netflow amount)
(d/dt amount netflow)

The redundant constraints are added since MISQ generates maximal QDEs. For example, since the M^+ constraint is transitive, if M^+ constraints hold between variables *a* and *b* and between *b* and *c*, MISQ would also include a redundant M^+ constraint between variables *a* and *c*.

3 Correctness and Uniqueness Theorem

A central feature of our method is that, given sufficient information on the input behaviors, it will generate a unique maximal QDE which is guaranteed to reproduce the input behaviors. This further implies that, if the user presents all system behaviors as input, we will produce a correct system model. This section presents the essential definitions and proves this central feature.

Consistent set of behaviors. A set of behaviors is consistent if it (potentially) represents a real physical system. This can be summarized by two criteria: First, relationships among variables must be qualitatively consistent among behaviors. In other words, if two variables are related by some constraint, then this constraint must be the same in all behaviors (e.g., not M^+ in one behavior and M^- in another). Second, dimensions must make sense (as they must in a real system). We might imagine a system in which a variable and its derivative have the same dimensions, and create behaviors for the system, but such a system could never actually exist.

Complete description. A description of a behavior is complete if three criteria are met. First, all variables in the system are identified. Second, values of the variables are given for all time points and intervals. Finally, dimensions are given for all variables. We do not require that all behaviors of a system be given. However, specifying too few behaviors may result in a model which is too constrained to produce behaviors of the system which were not given as input. The more behaviors that are given, the more constraints may be eliminated, thus making it less likely that the resulting model will be overconstrained and more likely that it is the intended model.

Theorem. Given a complete description of a consistent set of behaviors, we will produce the most constrained QDE which reproduces those behaviors. Furthermore, this QDE is unique.

Proof of Theorem. Given a fixed set of variables, two sets of constraints on these variables C_1 and C_2, and the behaviors consistent with these constraints $Beh(C_1)$ and $Beh(C_2)$, the set of behaviors consistent with both sets of constraints is given by the relation

$$Beh(C_1 \cup C_2) = Beh(C_1) \cap Beh(C_2) \quad (1)$$

Given a complete and consistent set of input behaviors, we exhaustively generate all constraints which are individually consistent with the behaviors. If we combine these constraints using (1), the intersection of their behavior sets will include all input behaviors. Thus, a correct model exists. Since we generate all consistent constraints, the resulting QDE is maximally constrained and unique.

4 Incomplete Information

Overview. The user may not always provide complete behavioral information. There are two ways in which behavioral information may be incomplete. First, entire variables may be missing from the behaviors. Second, information on variables may be partial in that their dimensions or some of their qualitative values are not given. Entire variables may be missing if the user does not know what set of variables are important to a system. Qualitative values may be omitted either when a variable is difficult to measure or when measurements are not available for all time points. Dimensions will generally be given for all variables specified by the user, but will be unavailable for variables created by MISQ.

If we have only partial information on some variables, constraints generated during the second phase of model building may be mutually inconsistent. We must eliminate these inconsistencies in order to generate a final model.

Once we have a consistent model, we check whether the model forms a connected graph. If it does not, either the behaviors describe independent processes or an essential system variable is missing. In this case, we consider adding new variables to the model.

Missing Qualitative Values and Dimensions. When qualitative values or dimensions are left unspecified, some generated constraints may make incompatible assumptions about the missing values. MISQ resolves this incompatibility by dropping one or more of the conflicting constraints. Since there is a choice of which constraints to delete, the resulting model is no longer unique. However, the model still reproduces the input behaviors.

One type of incompatibility arises with qualitative values. For example, suppose we have the constraints:

(d/dt *a b*) (M+ *a c*)

At a particular time-point, let the direction-of-change for *a* be unknown, the sign of *b* positive, and the direction-of-change of *c* decreasing. The constraints are mutually

inconsistent, since the derivative constraint assumes the direction-of-change of *a* to be increasing, while the M+ constraint assumes it to be decreasing.

Incompatibilities arising from missing dimensions are detected by an analysis which ensures that a set of constraints makes sense as a model of a physical system. For example, even without any dimensional information, we know that the following constraints are inconsistent:

(d/dt *a b*) (add *a b c*)

They are inconsistent because variables in an add constraint must have identical dimensions, but the dimensions of a variable and its derivative differ by a factor of 1/time.

As an example, we presented MISQ with the bathtub behavior in Figure 1, but with no dimensional information. MISQ generates six models. One of these is the desired model shown earlier. The others reflect the fact that, without dimensional information, MISQ is no longer able to distinguish between outflow and amount, as they are qualitatively indistinguishable in the specified behavior.

Missing Variables. Once we have a consistent set of constraints, we check to see whether they form a connected graph. If so, we consider our model complete. If not, there are two possibilities: we may be missing one or more variables, where the constraints associated with those variables would connect the model, or the behaviors may describe multiple independent processes.

These two possibilities define a spectrum of choices. At one extreme, we can choose to always consider the processes independent. At the other extreme, we can always generate some sequence of intermediate variables to connect any set of processes. In this spectrum, we have chosen the following position: We assume that the user has omitted only a small number of variables, and therefore only connect isolated parts of the model if we can do so by introducing at most one intermediate variable for each connection. Any portions of the model which cannot be connected in this way we consider independent.

New variables are added by a method called *relational pathfinding*, which is part of the general-purpose learning system Forte. We give a brief description of this method here. A complete description may be found in [Richards and Mooney, 1992]. Relational pathfinding provides a natural way to introduce new variables into a model. It is based on the assumption that relational concepts can be represented by one or more fixed paths between the constants that define an instance of the relation. In the case of qualitative modeling, we are looking for paths, composed of constraints, which will join model fragments into a coherent whole.

The pathfinding method seeks to find these paths by successively expanding the paths leading from each known system variable. To expand a path, we try adding all possible constraints involving one new variable. The added constraint and existing variables restrict the possible behaviors of the new variable. We take the set of new variables generated for each model fragment and look for an intersection between them. An intersection occurs when two new variables have consistent restrictions placed on their behaviors. When we find an intersection, the intersection point becomes a new system variable and the constraints leading to it are added to the model.

While relational pathfinding potentially amounts to exhaustive exponential search, it is generally successful for two reasons. First, by searching from all model fragments simultaneously, we greatly reduce the total number of paths explored before we reach an intersection. Second, we limit the length of the missing paths and hence the depth of search.

An example of a model containing variables added by relational pathfinding is included in the following section.

5 Examples

We have run MISQ on a variety of common models, including the U-tube modeled by GOLEM [Bratko, Muggleton, and Varšek, 1991], a nonlinear pendulum, a system of two cascaded tanks, and a system of two independent bathtubs. The latter two are discussed in detail below.

The U-tube consists of two tanks connected by a pipe at the bottom. GOLEM required one positive behavior, one hand-tailored positive timepoint, and six hand-generated negative timepoints. MISQ produced a correct model using only the positive behavior given to GOLEM. The nonlinear pendulum is a simple second-order system. MISQ produces a correct model given the first few states of a single damped behavior.

Cascaded tanks. Cascading two tanks so that the drain from one provides the inflow to the next provides a more complex system than the u-tube. We ran MISQ on various types of input:

-- qualitative, quantitative, and high-resolution data
-- with and without missing variables

A graph of the high-resolution data for the amount variables is shown in Figure 2. In all cases with complete dimensional information, MISQ produced the model in Figure 3, which is exactly the one we would expect. The constraints are:

```
(constant inflow_a)
(add outflow_a netflow_a inflow_a)
(add outflow_b netflow_b outflow_a)
(d/dt amount_b netflow_b)
(d/dt amount_a netflow_a)
(M+ amount_a outflow_a)
(M- amount_a netflow_a)
(M+ amount_b outflow_b)
(M- outflow_a netflow_a)
```

When we omitted system variables, we selected those that a user might realistically forget. We supposed the user measured all the flows and amounts but did not realize that the calculated netflow for each tank would be important. We therefore provided MISQ with the same qualitative behaviors as above, but omitted the netflow variables. The standard model generation process, before

relational pathfinding, produces the constraints:

(constant inflow_a)
(M+ amount_a outflow_a)
(M+ amount_b, outflow_b)

Note that these constraints are not connected. Relational pathfinding finds the missing two variables and six constraints, and again produces the correct model.

Two tubs. As a test of our ability to identify independent processes, we presented MISQ with two behaviors of a system containing two independent bathtubs. The standard model generation process produces the model:

(M+ amount_a outflow_a)
(M+ amount_b outflow_b)
(d/dt amount_a netflow_a)
(d/dt amount_b netflow_b),
(add netflow_a outflow_a inflow_a)
(add netflow_b outflow_b inflow_b)

This model includes all constraints needed for the two tubs (note that neither inflow is constant). The model is not connected, and relational pathfinding tries to add new variables. It is unable to connect the two bathtubs with one intermediate variable, and the model remains unchanged.

6 Related Work

Machine learning. Our approach is similar to the generalizing half of the Version Space algorithm described in [Mitchell, 1982]. Mitchell presents a method of deriving logical descriptions from a series of examples. Given a set of examples of the concept of interest, Version Space constructs the most specific conjunctive expression which includes those examples. We construct the most constrained model (essentially a conjunction of constraints) which reproduces all the input behaviors.

Model building. GENMODEL [Coiera, 1989] is a system which constructs maximally constrained qualitative models from completely specified qualitative behaviors. MISQ uses the same method to generate its initial set of constraints. However, MISQ generates fewer constraints, since it performs dimensional analysis. GENMODEL does not process quantitative behaviors, work with incomplete information, or perform dimensional analysis.

In [Bratko, Muggleton, and Varšek, 1991], the learning system GOLEM is used to abduce qualitative models. Their method requires hand-generated negative information (i.e., examples of behaviors which the system does *not* exhibit), it does not completely implement the QSIM constraints (e.g., corresponding values are ignored), and it does not use dimensional information.

The dimensional analysis MISQ performs is similar to [Bhaskar and Nigam, 1990], which uses dimensions to derive qualitative relations. However, [Bhaskar and Nigam, 1990] requires dimensions to be stated in terms of predefined fundamental types, whereas we allow dimensions to be user-defined or even to remain unspecified.

[DeCoste, 1990] presents a system for maintaining a qualitative understanding of a dynamic system from

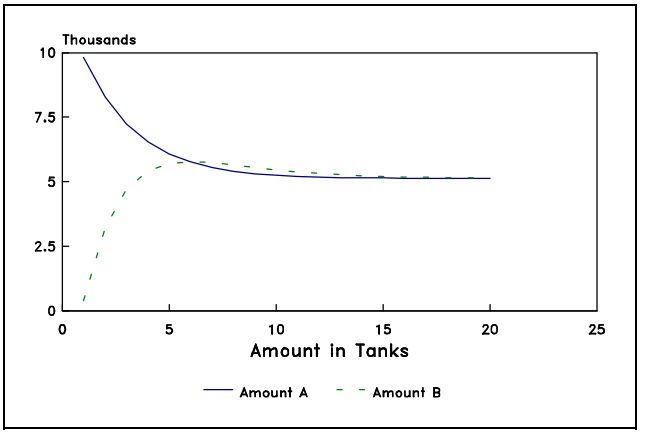

Figure 2. High-resolution quantitative data.

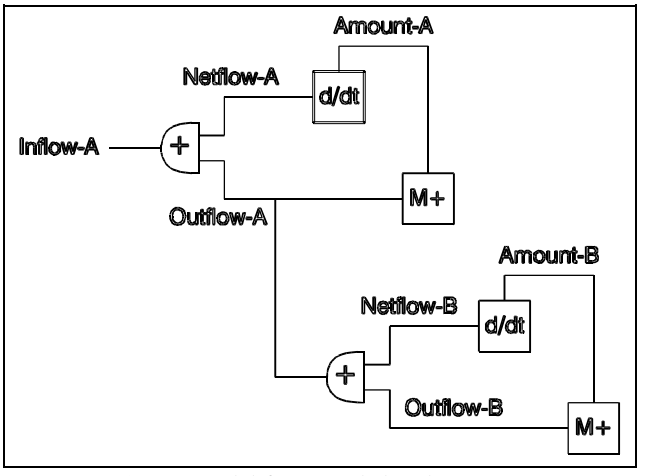

Figure 3. Correct model for cascaded tanks.

continuous quantitative inputs, but begins with a qualitative model. [Hellerstein, 1990] discusses the process of obtaining quantitative predictions of system performance in the absence of exact knowledge of the target system. And [Forbus and Falkenhainer, 1990] combines quantitative and qualitative models to produce "self-explanatory simulations," which produce quantitative predictions along with qualitative explanations of overall system behavior. But, again, Forbus and Falkenhainer require system models as input and exploit the relationship between the quantitative and qualitative models, rather than deriving the qualitative model from the input data.

7 Conclusions

Model building can be a difficult and time-consuming task. It can be simplified by automating some steps of the process. In this paper, we presented a method for automatically producing models from known behaviors. This approach is useful both in design and diagnosis.

In design, researchers often want models to produce specified quantitative or qualitative behaviors; our method can eliminate the need to handcraft these models. In diagnosis, our method can derive a model which reproduces a faulty behavior. Comparing the model of the faulty behavior with the correct model may show where the system fault lies. The fact that we can work directly with the available quantitative information is particularly helpful in this context.

There are several promising directions for further research. First, our approach can be extended to include other types of constraints like the QSIM *S* and *U* constraints. Second, when MISQ is given incomplete information and generates many potential models, additional filters could eliminate some of the proposed models. These filters could make use of behaviors which should not be produced by the model. Forte is already capable of using this type of negative information. Third, inconsistent input behaviors may represent a system which is crossing a transition. Modeling such a system would require constructing multiple models connected by well-defined transitions. Lastly, MISQ represents an extreme, knowledge-free approach to model-building. If more knowledge is available, for example in the form of a view-process library, this knowledge should be usable to restrict the set of possible constraints. Similarly, MISQ could be integrated with qualitative systems which work with partial quantitative information; rather than converting quantitative inputs to a purely qualitative model, we could retain the quantitative information and pass it, along with the model, to a system like Q2 ([Kuipers and Berleant, 1988]).

Acknowledgements

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 ITI-8905494, IRI-8904454, and IRI-9017047, by NASA grant NAG 9-512, and by the Texas Advanced Research Program under grand 003658-175.

References

R. Bhaskar and A. Nigam, "Qualitative Physics Using Dimensional Analysis," *Artificial Intelligence*, 45:73-111, 1990.

E. Coiera, "Generating Qualitative Models from Example Behaviors," Technical Report DCS 8901, Department of Computer Science, University of New South Wales, May 1989.

J. Crawford, A. Farquhar, and B. Kuipers, "QPC: A Compiler from Physical Models into Qualitative Differential Equations," *Proceedings of the Eighth National Conference on Artificial Intelligence* (AAAI-90), pp. 365-372, 1990.

D. DeCoste, "Dynamic Across-Time Measurement Interpretation," *Proceedings of the Eighth National Conference on Artificial Intelligence* (AAAI-90), pp. 373-379, 1990.

J. deKleer and J. 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," unpublished draft, 1990.

K. Forbus, "Qualitative Process Theory," *Artificial Intelligence*, 24:85-168, 1984.

K. Forbus, "The Qualitative Process Engine," Technical Report, Department of Computer Science, University of Illinois, 1986.

K. Forbus and B. Falkenhainer, "Setting Up Large-Scale Qualitative Models," *Proceedings of the Seventh National Conference on Artificial Intelligence* (AAAI-88), pp. 301-306, 1988.

K. Forbus and B. Falkenhainer, "Self-Explanatory Simulations: An integration of qualitative and quantitative knowledge," *Proceedings of the Eighth National Conference on Artificial Intelligence* (AAAI-90), pp. 380-387, 1990.

J. Hellerstein, "Obtaining Quantitative Predictions from Monotone Relationships," *Proceedings of the Eighth National Conference on Artificial Intelligence* (AAAI-90), pp. 388-394, 1990.

I. C. Kraan, B. L. Richards, and B. Kuipers, "Automatic Abduction of Qualitative Models," *Proceedings of the Fifth International Workshop on Qualitative Reasoning about Physical Systems*, 295-301, 1991.

B. Kuipers, "Qualitative Simulation," *Artificial Intelligence*, 29:289-338, 1986.

B. Kuipers, "Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge," *Automatica*, 25:571-585, 1989.

B. Kuipers, *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*, unpublished draft, 1990.

B. Kuipers and D. Berleant, "Using Incomplete Quantitative Knowledge in Qualitative Reasoning," *Proceeding of the Seventh National Conference on Artificial Intelligence* (AAAI-88), pp. 324-329, 1988.

T. M. Mitchell, "Generalization as Search," *Artificial Intelligence*, 18:203-226, 1982.

B. L. Richards and R. J. Mooney, "Learning Relations by Pathfinding," *Proceedings of the Tenth National Conference on Artificial Intelligence* (AAAI-92), 1992.

Notes

[1]A *behavior* is a continuous time-ordered sequence of variable values.

[2]We do not address issues of precision or noise. Our emphasis is qualitative model building, and in a realistic application we would expect sensor data to be pre-processed by a system designed to deal with these problems.

[3]A *qspace* is a totally ordered set of landmarks. *Landmarks* are values which break the domain of a variable into qualitatively distinct intervals. For example, the qspace of the temperature of a pot of water might be {*absolute-zero*, *freezing*, *boiling*, *infinity*}.

[4]The constraint (add x y z) means $x + y = z$, (d/dt x y) means $dx/dt = y$, (M^+ x y) means a strictly increasing monotonic function holds between x and y, and so forth.