Appears in *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*
pp. 965-970, Portland, OR, August 1996

Qualitative Multiple-Fault Diagnosis of Continuous Dynamic Systems Using Behavioral Modes

Siddarth Subramanian
National Instruments – Georgetown
1978 S. Austin Ave.
Georgetown, Texas 78626
sid@georgetown.com

Raymond J. Mooney
Dept. of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
mooney@cs.utexas.edu

Abstract

Most model-based diagnosis systems, such as GDE and Sherlock, have concerned discrete, static systems such as logic circuits and use simple constraint propagation to detect inconsistencies. However, sophisticated systems such as QSIM and QPE have been developed for qualitative modeling and simulation of continuous dynamic systems. We present an integration of these two lines of research as implemented in a system called QDOCS for multiple-fault diagnosis of continuous dynamic systems using QSIM models. The main contributions of the algorithm include a method for propagating dependencies while solving a general constraint satisfaction problem and a method for verifying the consistency of a behavior with a model across time. Through systematic experiments on two realistic engineering systems, we demonstrate that QDOCS demonstrates a better balance of generality, accuracy, and efficiency than competing methods.

Introduction

In a world increasingly filled with devices that exhibit complex dynamic behavior, online diagnostic systems are becoming increasingly important. To address this problem, researchers have devised various solutions over the last two decades (Shortliffe & Buchanan 1975; de Kleer & Williams 1987). These systems have been applied to the problems of medical diagnosis, as well as to combinational circuit diagnosis and similar domains. However, as we shall see, these diagnosis approaches are not directly suited to the kinds of continuous dynamic systems that we are interested in.

Traditional modes of reasoning about physical systems use differential equations to model their dynamics. However, these techniques are limited in their ability to usefully model large systems because of the difficulties in constructing accurate formulations of large systems, and because of the computational complexities involved in solving large systems of differential equations. One solution to this problem is to use *Qualitative Reasoning* (Forbus 1984; Kuipers 1984). Our work uses QSIM (Kuipers 1994) as the modelling language and applies a very general diagnostic technique to models described in this language.

Previous approaches to diagnosing faults in systems described with QSIM models have been limited in scope and have been unable to work with fault modes (Ng 1990; Lackinger & Nejdl 1991) or have made a single-fault assumption (Dvorak 1992). Most previous work on model-based diagnosis (Reiter 1987; de Kleer & Williams 1987) has concentrated on static systems and is generally insufficient to diagnose continuous dynamic systems. Few of the other approaches to diagnosis of continuous systems (Oyeleye, Finch, & Kramer 1990; Dague *et al.* 1991) have made use of a general modelling language such as that provided by QSIM or used any of the general diagnostic formalisms introduced by Reiter or DeKleer.

This work[1] is an integration of the two paradigms of model-based diagnosis and qualitative reasoning into a general, multiple-fault diagnosis system for continuous dynamic systems using behavioral modes with *a priori* probabilities. The diagnostic architecture is similar to SHERLOCK (de Kleer & Williams 1989) and the algorithm builds on INC-DIAGNOSE (Ng 1990). The system uses a general constraint-satisfaction technique to detect faults and trace dependencies in order to generate conflicts and diagnoses. A QSIM-based simulation component is used to verify hypotheses and detect additional inconsistencies. The implemented system, QDOCS (Qualitative Diagnosis Of Continuous Systems), is powerful enough to accurately diagnose a number of different faults in the Space Shuttle's Reaction Control System and a simple chemical reaction tank.

An Example

An example used to illustrate the algorithm consists of a simple bathtub with a drain. It is assumed that the bathtub is monitored by sensors measuring the

[1] A much more detailed account of this work can be found in (Subramanian 1995).

amount of water in the tub and the flow rate of the water through the drain. Some of the faults that can be posited about this system include a blocked drain, leaks in the tank, and sensors stuck at various levels.

This system is described using a qualitative differential equation or a QDE. A QDE is a set of constraints, each of which describes the relationship between two or more variables. For instance, an `M+` relation is said to exist between two variables if one is a monotonically increasing function of the other. So, in our normal bathtub model, there is an `M+` relation between the amount and the level of water in the bathtub and also between the the level and pressure, and the pressure and outflow rate. However, in a model of a blocked bathtub, the outflow rate is zero, and it is described by the constraint `ZERO-STD`.

The use of discrete mode variables in QSIM allows us to combine normal and faulty models of a system into a single description as shown here:

```
(M+ amount level)
(M+ level pressure)
(mode (drain-mode normal) (M+ pressure outflow))
(mode (drain-mode blocked) (ZERO-STD outflow))
(ADD netflow outflow inflow)
(D/DT amount netflow)
(CONSTANT inflow if*)
```

Here, the variable `drain-mode` takes on the possible values of *normal*, *blocked*, or *unknown* and the constraints shown above correspond to the two known modes of the bathtub's behavior.

For the purposes of diagnosis, these mode variables can then be associated with components of the system and their different values with behavioral modes of the component. Each of these behavioral modes has an *a priori* probability specified by the model-builder. The component structure used to represent the bathtub looks like this:

```
(defcomponents bathtub
 (drain drain-mode (normal 0.89)(blocked 0.1)
   (unknown 0.01))
 (levelsensor levelsensor-mode (normal 0.79)
  (stuck-at-0 0.1)(stuck-at-top 0.1)
  (unknown 0.01))
 (flowsensor flowsensor-mode (normal 0.79)
  (stuck-high 0.1)(stuck-at-0 0.1)(unknown 0.01))
 (inletvalve inletvalve-mode (normal 0.79)
  (stuck-closed 0.1)(unknown 0.01)))
```

Here, each entry consists of the component name (e.g., `drain`), the mode variable (`drain-mode`) and a list of behavioral modes with their *a priori* probabilities (`(normal 0.89) (blocked 0.1) (unknown 0.01)`).

The input to the diagnostic algorithm consists of a behavior, which is a sequence of qualitative values for a subset of the variables corresponding to sensor readings. The output of the algorithm is an assignment of values to the mode variables such that the resulting model is consistent with the observed behavior, i.e., the behavior corresponds to a QSIM simulation of the model.

As an example, suppose QDOCS is given the following single set of sensor readings from a behavior of the bathtub: `(level-sensed (0 top))`, `(outflow-sensed 0)` (i.e., the level sensed is somewhere between 0 and top and the outflow sensed is 0). This is clearly inconsistent with the normal model of the system which would predict a flow through the drain. Some of the valid diagnoses for this behavior include `[(drain-mode blocked)]`, `[(flowsensor-mode stuck-at-0)]` and `[(drain-mode blocked) (flowsensor-mode stuck-at-0)]`.

The above example motivates an approach of applying QSIM's constraint satisfaction techniques to detect inconsistencies between the sensor readings and the model. However, since the systems under study are dynamic systems that maintain temporal consistency, satisfying the constraints for a given set of sensor readings does not guarantee that the sequence of readings is consistent. The approach we discuss in the next section includes using the continuity checking of QSIM to check this temporal consistency.

QDOCS's Diagnostic Approach

QDOCS uses a standard diagnostic approach similar to that of (de Kleer & Williams 1989) and combines it with a hypothesis checker (and conflict generator) based on the QSIM algorithm of (Kuipers 1994).

Diagnosis Construction: Like SHERLOCK's technique for constructing diagnoses, QDOCS uses a best-first search mechanism and focusses its search on the leading candidate diagnoses as determined by their *a priori* probabilities. QDOCS maintains an *agenda* of hypotheses to be tested and a list of conflict sets. The former is initialized to the single hypothesis that everything is functioning normally while the latter is initialized to the null set.

The hypothesis checker is first called with the initial hypothesis of all the components being normal. If it returns a null value, the given behavior is consistent with the hypothesis; in other words, the given behavior is a possible result of running QSIM on the model assuming all component mode variables are in the `normal` mode. If there is no QSIM simulation that results in the given behavior, the checker returns a conflict set of component mode variable values. This conflict set is then added to the set of conflict sets, and the agenda is expanded by adding all hypotheses generated by changing the mode value of a single component in such a way that it *hits*[2] all the conflict sets. This process is repeated until one or more hypotheses are found to be consistent with the observations.

Checking Hypotheses: Most diagnostic systems like GDE (de Kleer & Williams 1987) use simple constraint propagation to determine conflict sets. However, QSIM requires a more complete constraint sat-

[2] A conflict is *hit* by a hypothesis if some literal in the diagnosis contradicts a literal in the conflict.

isfaction algorithm since a qualitative constraint typically does not entail a unique value for a remaining variable when all its other variables have been assigned. An earlier attempt to use QSIM to track dependencies for diagnosis (Ng 1990) only used a simple propagator. Since the propagator alone is not complete, Ng's program, INC-DIAGNOSE is not guaranteed to detect all inconsistencies.

QSIM takes a set of initial qualitative values for some or all of the variables of a model and produces a representation of all the possible behaviors of the system. The inputs to QSIM are 1) a qualitative differential equation (QDE) represented as a set of variables and constraints between them, and 2) an initial state represented by qualitative magnitudes and directions of change for some of these variables. QSIM first completes the state by solving the constraint satisfaction problem (CSP) defined by the initial set of values and the QDE. For each of the completed states satisfying the constraints, QSIM finds qualitative states that are possible successors and uses constraint satisfaction to determine which of these are consistent. The process of finding successors to states and filtering on constraints continues as QSIM builds a tree of states called a behavior tree.

There are two possible ways in which the QDE corresponding to a hypothesis can be inconsistent with a given set of sensor readings: 1) a particular set of readings may be incompatible with the QDE, or 2) all the sets of readings may be compatible with the QDE but the sequence may not correspond to any particular behavior in a QSIM behavior tree. QDOCS's approach is to first test for consistency between individual sets of readings and the QDE by using QSIM's CSP algorithms, and then, test to see if the model fits the sequence, i.e., if the sequence of readings corresponds to a behavior generated by QSIM.

For the first step, QDOCS modifies QSIM's constraint satisfier to keep track of mode-variables whose values played a role in reducing the set of possible values for a variable. Each variable and constraint is associated with an initially empty dependency set of mode variables. Whenever a constraint causes a variable's set of possible values to decrease, the dependency set of the variable is updated with the union of its old dependency set, the dependency set associated with the constraint, and the mode variable, if any, that is associated with the constraint. When a variable reduces the set of possible tuples associated with the constraint, the constraint's dependency set is similarly updated with the union. When a variable is left with no possible values, its current dependency set is returned as a conflict set.

QDOCS's approach to solving the CSP, based on QSIM's, is to first establish node consistency by ensuring that each constraint is satisfied by the possible values of the variables it acts upon, and then use Waltz filtering (Waltz 1975) to establish arc consistency, by propagating the results of the node consistency checker to other variables and constraints. Finally, QDOCS uses backtracking to assign values to variables. The first step above is a standard constraint propagation algorithm as used in traditional diagnostic systems while the last two steps will be referred to as the constraint satisfaction algorithm of QDOCS. Mode variable dependencies are maintained at each stage of this process so that the procedure can stop if an inconsistency is detected at any step. The Waltz filtering step is performed incrementally and at each point selects the most restrictive constraint (i.e., the one most likely to fail) to process and propagates its effect on the rest of the network. This heuristic of first filtering on the most restrictive constraints helps reduce the size of conflict sets since the most restrictive constraints are those with the least number of initial possible tuples, and therefore are more likely to lead to an inconsistency.

For the second part of the algorithm, QDOCS must track a QSIM simulation and match all possible successors at each stage of the simulation with the given sensor readings. Successors that do not either match the current set of sensor readings or the next set of sensor readings in the observed sequence are pruned out. Whenever the computed states corresponding to a particular set of sensor readings fail to have any successors matching the next set of sensor readings, an inconsistency is noted and the entire hypothesis is returned as a conflict.

This last step differs from the general QDOCS approach of trying to isolate the individual mode variable values responsible for an inconsistency. We discovered through our experiments (Subramanian 1995) that keeping track of the dependencies of variable values on mode variables across time was computationally expensive while giving us little benefit as most conflict sets were still almost as large as the entire hypothesis set. This kind of inconsistency was much rarer than inconsistencies in individual states detected through either the propagation or constraint satisfaction phases of the hypothesis checker.

Experiments

The experiments presented in this section test three primary claims about QDOCS. First, because it can detect inconsistencies and generate conflicts when propagation is blocked, QDOCS is more accurate than an approach that only uses propagation such as INC-DIAGNOSE. Second, because it uses dependencies and conflict sets to focus diagnosis, QDOCS is more efficient than a baseline generate-and-test approach. Third, each of the phases of QDOCS's hypothesis checking algorithm contributes to improving its accuracy or efficiency.

Experimental Methodology: In each of our domains, the *a priori* probabilities in the model were used to randomly generate sets of multiple faults. QSIM was used to simulate the model corresponding to these mul-

tiple fault hypotheses, and a behavior randomly chosen from the resulting behavior tree was used to test QDOCS.

QDOCS was compared to various different techniques and for each of these we collected data on the efficiency and accuracy of the methods. First, a generate-and-test method was used as a baseline comparison. This technique used the same hypothesis checker as QDOCS, and simply tests hypotheses generated in most-probable-first order until one or more hypotheses are found to be consistent with the observations. Note that given the fact that QSIM makes acausal inferences, a generate-and-test procedure is the best we can do without using QDOCS-style dependency propagation.

We also compared QDOCS with a number of ablated versions in order to justify all the different parts of the hypothesis checker. First, it was compared against a system that simply used QDOCS's constraint propagation procedure which is equivalent to INC-DIAGNOSE (enhanced to handle behavioral modes). Another ablated version of QDOCS we test against is one with both the propagation and constraint satisfaction parts of the code but without across-time verification. This test is to determine if the across-time verifier (which is one of the most computationally expensive parts of QDOCS) is worthwhile in improving the accuracy of the system. Finally, we test a version of QDOCS that used the constraint satisfaction and across-time verification portions of the hypothesis checker but skipped the constraint propagation portion. This comparison was run to verify that that the constraint propagation algorithm speeds up the constraint satisfaction process even though the constraint satisfaction and across-time verification algorithms together are just as powerful (in terms of accuracy of diagnoses) as the complete QDOCS hypothesis checker.

On each problem, the tested technique was run until the best remaining hypothesis had a probability of less than a tenth of the probability of the best (i.e., most probable) hypothesis that was found to be consistent with the observations thus far. This would give us a range of all the consistent hypotheses that were within an order of magnitude of each other in *a priori* probability and would provide a termination condition for the top-level procedure of QDOCS. In each of our domains, we first generated a test suite of 100 examples and ran the above experiments on all of them.

Reaction Control System: The first problem we look at is that of diagnosing faults in the Reaction Control System (RCS) of the Space Shuttle. The RCS is a collection of jets that provides motion control for the orbiter when it is in space. These jets are fired appropriately whenever changes need to be made to the orientation or position of the craft. Detailed descriptions of this problem domain and our approaches to it can be found in (Subramanian 1995).

A QSIM model for this system was first built by (Kay 1992). This model has been extended and modified by us for the purposes of diagnosis. The complete QSIM model contains 135 constraints and 23 components, each with multiple behavioral modes. Some of the kinds of faults modeled include pressure regulators stuck open and closed, leaks in the helium tank, the fuel tank, or the fuel line, and sensors being stuck low or high.

Method	Avg. # Hyps.	Most Prob. Corr.%	Member Subs.%	Run Time (sec)	Hyps. Tested
Gen & Test	1.39	77.00	100.00	1288.77	456.55
Prop. only	2.91	29.00	85.00	44.39	19.71
No Across	1.71	42.00	84.00	85.68	25.29
No Prop.	1.39	77.00	100.00	647.95	52.62
Qdocs	1.39	77.00	100.00	454.02	52.70

Figure 1: Results in the RCS domain

Since the actual probabilities of the faults were unknown, they were assigned by us with normal modes being much more common than the fault modes. As with all QDOCS models, we make the assumption that the faults are independent of each other.

We ran the series of experiments described above on the RCS system. The results are summarized in Figure 1. The first column reports the average number of hypotheses generated per diagnosis problem for each of the tests. The second and third columns show different measures of accuracy for each method, while the last two columns show different measures of efficiency.

For each method we separated out the most probable hypotheses (often more than one if there were a few equally probable hypotheses) and compared these to the correct hypothesis. The percentage of cases where the correct hypothesis was among these is reported in the second column. In many cases, a subset of the correct faults is sufficient to model the given behavior. The third column shows the percentage of cases in which some hypothesis is a subset of the faults of the correct hypothesis. The last two columns show respectively the average time taken for each problem on a Sparc 5 workstation running Lucid Common Lisp and the number of hypotheses the hypothesis checker actually had to test.

When we compare the generate-and-test method (first line) to the complete QDOCS algorithm (last line), we see that they both have identical accuracies – in 77% of the cases the correct solution was among the most probable. This result is as expected since a systematic elimination of hypotheses as in the generate and test method is guaranteed to reach the right hypotheses eventually. The big difference appears in the average number of hypotheses tested – the generate and test method tests 8.7 times more hypotheses than QDOCS. This shows that QDOCS is able to narrow the search space considerably using its dependency propagation algorithms but the ratio of run times, which is 2.8 to 1 in favor of QDOCS, indicates that there is a

Method	Avg. # Hyps.	Most Prob. Corr.%	Member Subs.%	Run Time (sec)	Hyps. Tested
Gen & Test	4.96	50.00	98.00	59.21	215.14
Prop. only	4.63	39.00	99.00	6.63	26.60
No Across	4.65	39.00	99.00	7.69	26.83
No Prop.	4.96	50.00	98.00	31.50	24.31
QDOCS	4.96	50.00	98.00	27.39	33.95

Figure 2: Results in the Level-Controller domain

cost to be paid for this. This is still a substantial advantage for QDOCS over the simpler generate and test method.

Figure 1 also shows the results of the ablation tests. We find that using just propagation or propagation and constraint satisfaction reduces the accuracy of the method since we are not verifying the hypotheses across time, while leaving out propagation has no effect on accuracy (compared to QDOCS) but the propagation step does speed up the process of finding contradictions and hence the overall computation time.

Another interesting experiment we conducted regarding run time comparisons between QDOCS and the generate and test method was a study of a part of the RCS subsystem consisting of a single propellant flow path to the thruster. The model for this system is almost exactly half the size of the full RCS subsystem model. We generated problems in the same way as for the experiments reported on in Figure 1, and ran 100 problems through the generate and test and QDOCS algorithms. The accuracies were identical (86% correct, 100% subset) between the two methods. However, the run times averaged 264 seconds for the generate and test and 221 for QDOCS. This is a ratio of only 1.2 to 1 even though the ratio of hypotheses tested was 4.4 to 1. The corresponding ratios for the complete system are 2.9 to 1 and 8.7 to 1. This suggests that for similar problems, the larger the problem size, the greater the advantage of using a dependency propagation algorithm like QDOCS to generate conflict sets. We therefore expect the advantages of QDOCS to be greater for even larger problem sizes.

Level-Controlled Tank: We studied one other system, a level controller for a reaction tank, taken from a standard control systems textbook, (Kuo 1991). The main reason this system is of interest is to show that the QDOCS mode of dependency propagation is useful even for feedback systems. Some researchers (e.g., (Dvorak & Kuipers 1992)) have held that such an algorithm would not be useful in dynamic systems with feedback loops because variable values are usually dependent on all constraints.

The level-controlled tank is modeled using a QSIM model with 45 constraints and a component structure with 14 components. We ran all the experiments described in the methodology section on this model of the controlled tank. The results are summarized in Figure 2.

As in the equivalent experiments with the RCS, QDOCS does better than the other techniques. It is about 2.2 times faster and tests 6.3 times fewer hypotheses than the generate and test method. QDOCS is also more accurate than either propagation alone or propagation and constraint satisfaction.

Future Work

This work needs to be further extended and applied to a variety of different engineering systems. One important first step towards applying such a system is to integrate it with a monitoring system such as MIMIC (Dvorak 1992). This would require the use of semi-quantitative information which is likely to add more power to QDOCS.

One area we have investigated but which could use further research is that of efficient caching of possible values of different variables during the constraint satisfaction phase of the algorithm. Traditional truth maintenance systems like the ATMS(de Kleer 1986) are not useful for this purpose since the range of possible values for a variable is rarely narrowed to a single one. Initial results on our attempt at implementing a more general caching mechanism are reported in (Subramanian 1995) but these are somewhat discouraging in that the overheads required to maintain the caches in our implementation are often higher than the computational savings. Further investigation will be required to formulate a truly efficient caching scheme.

Related Work

Compared to QDOCS, the previous diagnosis systems for QSIM models all have important limitations. INC-DIAGNOSE (Ng 1990) was an application of Reiter's theory of diagnosis (Reiter 1987) to QSIM models. Its main limitations were that first, like Reiter's theory, it was restricted to models where no fault mode information was known, and second, it used a constraint propagator that was not guaranteed to detect all inconsistencies. Another system that used the INC-DIAGNOSE approach in the context of a monitoring system is DIAMON(Lackinger & Nejdl 1991). Again, due to its dependence on the simple constraint propagation in INC-DIAGNOSE, it is only able to detect a small subset of possible faults which QDOCS can diagnose.

The other previous diagnosis work on QSIM models, MIMIC (Dvorak 1992), has several limitations. First, MIMIC requires the model builder to provide a structural model of the system in addition to the QSIM constraint model. This structural model was fixed and could not change under different fault models. QDOCS does not require this since it uses a constraint-satisfaction algorithm to determine the causes for inconsistencies. Second, MIMIC uses a very simple dependency tracing algorithm to generate potential single-fault diagnoses. This algorithm looks at the structural graph from the point at which the fault is detected and considers all components it finds upstream

as possible candidates for failure and thus generates a larger set of possible component failures.

A number of other researchers have looked at diagnosis in the context of monitoring continuous systems (Oyeleye, Finch, & Kramer 1990; Doyle & Fayyad 1991). Each of these systems concentrates on different aspects of the monitoring process, but none performs multiple-fault diagnosis using behavioral modes.

Some recent work by Dressler (Dressler 1994) performs model-based diagnosis on a dynamical system (a ballast tank system) using a variant of GDE. It first reduces the model to a version suitable for constraint propagation, and then considers only conflicts generated by constraints acting at a particular time. While this is an efficient technique that apparently works well for their application, it is not a general method since some systems may have faults which can only be detected using information gathered across time.

Conclusion

We have described an architecture for diagnosing systems described by qualitative differential equations that performs multiple-fault diagnosis using behavioral modes. An implemented system, QDOCS, has been shown to be powerful enough to accurately generate diagnoses from qualitative behaviors of a fairly complex system – the Reaction Control System of the Space Shuttle. The approach is more powerful than previous methods in that it uses 1) a general modelling framework (QSIM), 2) a more complete diagnostic architecture and 3) a more complete constraint-satisfaction algorithm as opposed to simple propagation.

References

Dague, P.; Jehl, O.; Deves, P.; Luciani, P.; and Taillibert, P. 1991. When oscillators stop oscillating. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 1109–1115.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32:97–130.

de Kleer, J., and Williams, B. C. 1989. Diagnosis with behavioral modes. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1324–1330.

de Kleer, J. 1986. An assumption-based TMS. *Artificial Intelligence* 28:127–162.

Doyle, R. J., and Fayyad, U. M. 1991. Sensor selection techniques in device monitoring. In *Proceedings of the Second Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, 154–163. IEEE Computer Society Press.

Dressler, O. 1994. Model-based diagnosis on board: Magellan-MT inside. In *Fifth International Workshop on Principles of Diagnosis*, 87–92.

Dvorak, D., and Kuipers, B. 1992. Model-based monitoring of dynamic systems. In Hamscher, W.; Console, L.; and de Kleer, J., eds., *Readings in Model-Based Diagnosis*. San Mateo, CA: Morgan Kaufmann. 249–254.

Dvorak, D. 1992. *Monitoring and Diagnosis of Continuous Dynamic Systems Using Semiquantitative Simulation.* Ph.D. Dissertation, University of Texas, Austin, TX.

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

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

Kuipers, B. J. 1984. Commonsense reasoning about causality: Deriving behavior from structure. *Artificial Intelligence* 24:169–203.

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

Kuo, B. C. 1991. *Automatic Control Systems*. Engelwood Cliffs, New Jersey: Prentice Hall.

Lackinger, F., and Nejdl, W. 1991. Integrating model-based monitoring and diagnosis of complex dynamic systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 1123–1128.

Ng, H. T. 1990. Model-based, multiple fault diagnosis of time-varying, continuous physical devices. In *Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications*, 9–15.

Oyeleye, O. O.; Finch, F. E.; and Kramer, M. A. 1990. Qualitative modeling and fault diagnosis of dynamic processes by Midas. *Chemical Engineering Communications* 96:205–228.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32:57–95.

Shortliffe, E., and Buchanan, B. 1975. A model of inexact reasoning in medicine. *Mathematical Biosciences* 23:351–379.

Subramanian, S., and Mooney, R. J. 1995. Multiple-fault diagnosis using qualitative models and fault modes. In *IJCAI-95 Workshop on Engineering Problems in Qualitative Reasoning.*

Subramanian, S. 1995. *Qualitative Multiple-Fault Diagnosis of Continuous Dynamic Systems Using Behavioral Modes.* Ph.D. Dissertation, Department of Computer Science, University of Texas, Austin, TX. Also appears as Artificial Intelligence Laboratory Technical Report AI 95-239 and at URL ftp://ftp.cs.utexas.edu/pub/mooney/papers/qdocs-dissertation-95.ps.Z.

Waltz, D. 1975. Understanding line drawings of scenes with shadows. In Winston, P. H., ed., *The Psychology of Computer Vision*. Cambridge, Mass.: McGraw Hill. 19–91.