Model-Based, Multiple-Fault Diagnosis of Dynamic, Continuous Physical Devices

Hwee Tou Ng

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

December 1991

As technology advances, more and more complex devices and systems are designed and built. Consider electronic circuits, nuclear-power plants, petrochemical refineries, and space shuttle launch processing systems. Because of their complexity, these devices are difficult to diagnose when they malfunction. Research in automated diagnosis addresses this problem by constructing computerized diagnostics that can pinpoint faults based on a device's observed misbehavior.

Traditional expert-system technology can be applied to automated diagnosis, as illustrated by successful medical diagnostic programs such as Mycin and Internist. This approach uses production rules that directly relate symptoms to the possible underlying causes.

An alternative approach to diagnosis is based on an underlying model of a device's structure and behavior [1, 2]. Device diagnosis from first principles uses a model of a physical device to explain discrepancies between its faulty and normal behavior. This model-based approach has some advantages over the associational, heuristic-rule approach of conventional expert systems. It is compositional in that it lets us define models for a library of basic components, and it works on all systems composed from those components. The system designer can focus on getting the component models right, leading to more robust and sound diagnosis systems. The potential is also better for verification of the underlying knowledge base.

Modifying the algorithm

My approach builds on Reiter's theory of diagnosis [2] and Kuipers' qualitative simulation [3]. Reiter proposed a formal theory of diagnosis from first principles that reasons from system descriptions and observations of system behavior. In particular, his algorithm computes all minimal diagnoses of a device, including multiple-fault diagnoses. However, like others, he only applied his theory to diagnosing digital circuits, which are representative only of physical devices with static, persistent states. To the best of my knowledge, none of the work in model-based diagnosis has looked at the applicability of Reiter's theory of diagnosis to physical devices such as those studied by Kuipers [3], in which continuous change is an important feature.

On the other hand, Kuipers suggested one way of applying his work on simulating continuous systems [3, 4] to the problem of diagnosis. He suggested incorporating a set of fault models simulated with Qsim, and comparing predicted behaviors against observations. The fault model whose predicted behavior matches the faulty behavior then determines the set of faults in the device. The drawback of this approach is that the set of fault models needs to be determined beforehand and explicitly specified. Unanticipated faults are not modeled and would go undetected. Furthermore, the number of fault models needed to handle all possible combinations of multiple faults would be prohibitively high.

My diagnosis algorithm, called Inc-Diagnose, overcomes these difficulties. It diagnoses dynamic, continuous devices of the kind modeled by Qsim. It does not need explicit fault models, and it handles multiple faults. A Qsim-style constraint model describes the physical device as a set of components. A set of Qsim qualitative states characterizes the device's observed faulty behavior over time. Using qualitative state representation and qualitative constraints transforms a problem of a continuous nature into a discrete one: Instead of dealing with continuous behavior, we examine a finite number of discrete qualitative states to diagnose possible faults.

Furthermore, the modified algorithm Inc-Diagnose makes Reiter's approach incremental. An implicit assumption in Reiter's algorithm is that all observations are available at the start of diagnosis, which is often not the case. Since taking measurements is generally expensive, we want to be able to intermix measurement with fault hypothesis generation, especially for dynamic, continuous devices. The modified approach is preferable because it generates fault hypotheses incrementally.

Reiter's theory of diagnosis

In Reiter's theory of diagnosis, a diagnosis problem consists of a system description (SD), a set of system components ($Comp$), and observations of the system (Obs). The system description specifies the general rules that must be obeyed for the system to function normally. In a logical representation, the system description is a set of first-order formulas, although it need not be; for example, in our current application, it is a set of constraints expressed in Qsim's constraint language.

Diagnosis. A diagnosis is defined to be a minimal set Δ of system components such that assuming all components in Δ are faulty and all components in $Comp - \Delta$ are normal is consistent with $SD \cup Obs$.

Under this definition, more than one diagnosis might exist in which none of the diagnoses is a subset of another. Diagnosis must be minimal (with respect to set inclusion), since only a small set of device components is usually faulty; that is, an "Occam's razor" criterion.

In addition, this definition requires a test of consistency, which is undecidable in first-order logic. This implies that finding a diagnosis in the general case is undecidable as well. However, in many practical situations, such as diagnosing time-varying devices described by qualitative differential equations, this consistency check is decidable.

In Reiter's formulation, a diagnosis only needs to be consistent with the system description and observations. This is in contrast with an abductive approach to diagnosis [6], in which the hypotheses about diseases (or faulty components in the case of device diagnosis) must imply or explain the symptoms. That is, Reiter's approach does not require a notion of causality, just consistency.

Based on the above definition, we can come up with a naive and straightforward generate-and-test method to compute all diagnoses: Systematically generate all possible subsets of $Comp$, starting with the smallest subset and testing each one for consistency. However, this approach leads to a combinatorial explosion of the search space and is highly inefficient.

Instead, Reiter proposed an algorithm based on the concept of a conflict set. His algorithm computes the set of all diagnoses more efficiently. (Given the undecidability problem, this is an "algorithm" only if the consistency-checking problem is assumed to be decidable.)

Conflict set. A conflict set C is a set of system components such that assuming that all components in C are normal is inconsistent with $SD \cup Obs$.

In other words, at least one of the components in a conflict set must be faulty. This implies that the intersection of any diagnosis Δ with any conflict set C must be nonempty. To see why, let's assume that for some diagnosis Δ and conflict set C, $\Delta \cap C = \emptyset$. Since C is a conflict set, this implies that it is inconsistent with $SD \cup Obs$ to assume that all components in C are normal. Since $\Delta \cap C = \emptyset$, $Comp - \Delta$ is a superset of C, and it is also inconsistent with $SD \cup Obs$ to assume that all components in $Comp - \Delta$ are normal. (If making a set of assumptions A leads to an inconsistency, then making more assumptions in addition to A is certainly inconsistent too.) This directly contradicts the hypothesis that Δ is a diagnosis.

If we can somehow compute all conflict sets, then the set of all diagnoses consists of all minimal sets such that for every diagnosis Δ, the intersection of Δ with every conflict set is nonempty. That is, a diagnosis "hits" every conflict set.

The Diagnose algorithm. Reiter's Diagnose algorithm is based on these observations, but it computes minimal diagnoses without necessarily computing all conflict sets. I present here the corrected version of Reiter's algorithm as described in Greiner et al. [5] The algorithm requires a consistency-checking module with the following property: Given a system description SD, observations Obs, and a subset $Comp'$ of the system components $Comp$, the module either returns a conflict set $C \subseteq Comp'$ if such a conflict set exists, or $\emptyset$ if no such C exists (that is, if it is consistent to assume that all components in $Comp'$ are normal). If SD and Obs are first-order logic formulas, the module will be a theorem prover that serves as a consistency checker. (In diagnosing dynamic devices, I used the Qsim constraint propagator as the module; more on this later.)

The Diagnose algorithm computes the set of all diagnoses by repeatedly calling the consistency-checking module with the appropriate subsets of system components, and it builds a directed acyclic graph, called a hitting-set DAG or HS-DAG, as the computation progresses. Each node in the HS-DAG is labeled with a conflict set, and each edge is labeled with a system component. We define the path label of a node to be the set of edge labels (corresponding to system components) from the root of the HS-DAG to the node.

The algorithm builds an HS-DAG as follows:

1. At the root node, call the consistency checker with the set of all system components $Comp$. Label the root node with the conflict set returned by the module. The root node's path label is set to $\emptyset$.

2. For every node n labeled with a conflict set $C = \{c_1, c_2, \ldots, c_k\}$, sprout k child nodes from node n, where each child node n_{c_i} is connected to its parent node n with an edge labeled by the component c_i. For every child node n_{c_i}, suppose its path label is the set P_i. To determine the appropriate node label of this child node, call the module with the system components $Comp - P_i$. If a conflict set C_i is returned, make it the label of this child node. If the module returns $\emptyset$, mark this child node as completed.

The Diagnose algorithm builds an HS-DAG, making calls to the consistency-checking module as necessary, until all DAG leaf nodes are completed.

Heuristic pruning. The resulting HS-DAG is such that the set of path labels of all completed leaf nodes is a superset of the set of all diagnoses. There are several pruning heuristics that are guaranteed to improve the algorithm's computational efficiency. Furthermore, after these heuristics have been applied, the set of all diagnoses is exactly the set of path labels of the completed leaf nodes.

The Diagnose algorithm includes these heuristic-pruning refinements:

- Generate the HS-DAG in a breadth-first, left-to-right order. This facilitates the use of the other pruning heuristics.

- If possible, reuse a previous conflict set found as the node label of the current node. Calling the module to check consistency is an expensive operation, and reusing a conflict set helps to reduce this cost.

- If the path label of the current node is a proper superset of the path label of some completed node found earlier, then close the current node. That is, ignore the current node and do not generate any successor nodes, since we are only interested in finding minimal diagnoses.

- If the path label of a child node n' of the current node contains the same set of components as the path label of some previous node n (which may or may not be completed already), then merge n' and n into one node (thus forming a DAG). Both paths lead to the same set of diagnoses, and merging them avoids duplicating work.

- Suppose the conflict set C returned by the module for the current node is a proper subset of the conflict set C' of some previous node n'. (This is possible, since the consistency checker only returns a conflict set, not necessarily a minimal conflict set.) Then for each edge labeled with a component in $C' - C$ sprouting from node n', remove the node connected by this edge and all its descendants, except those nodes with another ancestor that is not being removed. The node label of n' is changed to C. This is justified because, if the module had returned the smaller conflict set C at node n' when it was first called, C would have been the label for node n', and the redundant edges and the descendants would not have existed in the first place.

While the Diagnose algorithm is relatively efficient for computing all diagnoses, its performance will still be exponential in the worst case, since the diagnosis problem posed here is NP-hard. The set-covering problem [6], a well known NP-hard problem, can be reduced to the diagnosis problem formulated here [2].

Representing and diagnosing continuous devices

In my system, a continuous device is decomposed into a finite set of system components, and one or more qualitative differential equations governs the behavior of each component. This kind of device model is known as a component-connection model. The work on qualitative simulation [3, 4] proved useful in representing continuous devices and their behavioral states.

As an example, consider a simple temperature controller, whose schematic diagram is shown in Figure 1. The temperature controller consists of five components: a temperature sensor, a temperature control knob, a proportional controller, an on/off power switch, and an electric heating/cooling component.

The function of this device is to control room temperature T_i^{ob}. The device is connected to a power source with power P^{ob}. When the power switch is turned on (represented as $W = 1$) and the temperature T_s^{ob} set by the temperature control knob differs from the room temperature, heat flow HF is generated (in the form of hot or cold air, depending on the direction of temperature difference). Furthermore, the amount of heat flow generated is proportional to the temperature difference $T_s^{ob} - T_i^{ob}$.

Given the structure of the temperature controller and the constraints governing its correct behavior (for example, the proportionality constraint between the temperature difference and the heat flow generated), the diagnostic task is to identify failing components from a description of the device misbehavior. For instance, we might observe on a cold winter day that the device is connected to the power source, the power switch is turned on, the temperature set by the control knob is at a specific room temperature, but the temperature in the room falls significantly below the set room temperature, with no heat flow generated into the room.

Figure 2 lists the qualitative constraints on the device and the component affected by each constraint. The first constraint reflects the fact that if the temperature sensor is working normally, then the observed temperature in the room is the same as the temperature actually recorded by the sensor (T_i). The second constraint on the temperature control knob is similar. The third and fourth constraints require that the heat-flow adjustment amount (a) must be proportional to the difference (e) between the temperature set (T_s) and the temperature in the room (T_i). Qsim uses a "+" constraint to represent addition and (with the arguments appropriately rearranged) subtraction. The plus box in the figure is not an adder; it denotes the constraint on the three variables T_i, T_s, and e. In the fourth constraint, M^+ denotes a monotonically increasing function. The fifth constraint implies that if

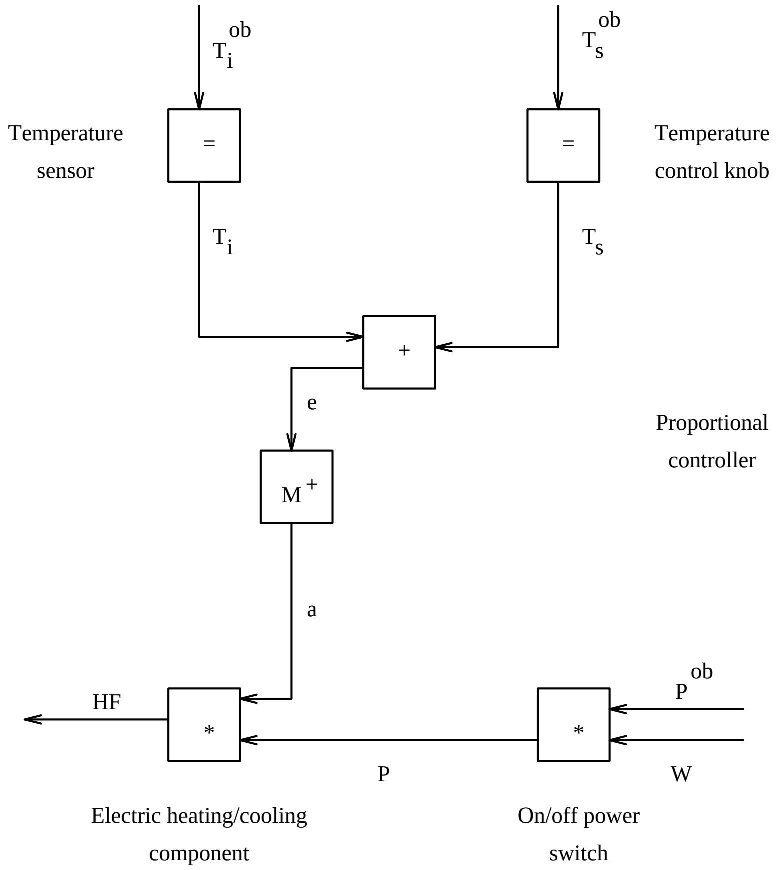

Figure 1: A temperature controller.

Qualitative constraint	Component affected
$T_i^{ob} = T_i$	Temperature sensor
$T_s^{ob} = T_s$	Temperature control knob
$T_i + e = T_s$	Proportional controller
$a = M^+(e)$	Proportional controller
$P^{ob} \times W = P$	On/off power switch
$a \times P = HF$	Electric heating/cooling component

Figure 2: Qualitative constraints on the device.

	s_0	s_1	s_2	s_3	s_4
T_i^{ob}	T_{rm},Dec	(Cold,T_{rm}),Dec	Cold,Std	Cold,Std	Cold,Std
T_s^{ob}	T_{rm},Std	T_{rm},Std	T_{rm},Std	T_{rm},Std	T_{rm},Std
P^{ob}	P_{sup},Std	P_{sup},Std	P_{sup},Std	P_{sup},Std	P_{sup},Std
W	1,Std	1,Std	1,Std	1,Std	1,Std
HF	0,Std	0,Std	0,Std	0,Std	0,Std
T_i	T_{rm},Dec	(T_{mid},T_{rm}),Dec	T_{mid},Dec	(Cold,T_{mid}),Dec	Cold,Std
a	0,Std	0,Std	0,Std	0,Std	0,Std
T_s	T_{rm},Std	T_{rm},Std	T_{rm},Std	T_{rm},Std	T_{rm},Std

Table 1: Qualitative states of the faulty behavior of the temperature controller.

the on/off power switch is functioning properly, the power input (P) to the electric heating/cooling component is the same as the product of the observed power input (P^{ob}) and the on/off control W (where $W = 1$ if the switch is on and $W = 0$ otherwise). The last constraint requires the heating/cooling component to output the correct amount of heat flow (HF) according to the product of the adjustment amount (a) and the power supply (P).

Qualitative states. As in Qsim, my system represents a qualitative state of a continuous device by the qualitative values of the system parameters. A qualitative value has two components: a qualitative magnitude and a qualitative direction. A qualitative magnitude can either be a landmark value or an open interval between two landmark values, where a landmark value is a value of special significance that a parameter takes on at some point in time. A qualitative direction can be increasing (Inc), decreasing (Dec), or steady (Std).

We can represent the faulty behavior of a continuous device over time as a finite set of qualitative states. For instance, Table 1 shows the observed faulty behavior of the temperature controller represented as five successive qualitative states $s_0, \ldots, s_4$. In the qualitative state s_0, the qualitative value of the room temperature is (T_{rm},Dec), where T_{rm} is the qualitative magnitude and Dec is the qualitative direction.

We assume that the input qualitative states are well formed and conform to the necessary across-state continuity constraints, but we could use Qsim to check for such continuity constraints of the qualitative states.

Constraint propagation. Given that we have used a set of qualitative constraints as the system description, and a set of qualitative states as the observations, it follows naturally that we use the Qsim constraint-propagation module as the consistency-checking module required by Reiter's algorithm.

To be more precise, when a call is made to the Qsim constraint-propagation module with a state s and a subset $Comp'$ of system components, the constraint propagator removes all but the constraints related to the set of components $Comp'$. Constraint propagation then tries to propagate parameter values through the remaining constraints as much as possible. Each time the module successfully propagates a value via some constraint, the module flags that constraint as "used". If at some point during propagation a parameter value takes on two distinct qualitative magnitudes or directions, an inconsistency is detected. The propagation module

For each of the successive states $s_0, \ldots, s_{n-1}$
/* Process state s_i */
Open all the leaf nodes in the current HS-DAG
(mark them as yet to be explored)
For each of the leaf nodes in the order of increasing depths
Make the same call to the consistency-checking module as in
Reiter's algorithm but use s_i as the new Obs
Expand the HS-DAG as in Reiter's algorithm
The diagnoses are all path labels of the completed leaf nodes in the final HS-DAG

Figure 3: The Inc-Diagnose algorithm.

then returns a conflict set containing all components that have at least one constraint flagged as "used". These components form a conflict set, since enforcing their constraints would lead to a contradiction. However, if constraint propagation halts without detecting any inconsistency, the propagation module returns the empty set. That is, it is consistent to assume that all components in $Comp'$ are normal.

Since constraint propagation is incomplete (some constraints can be blocked during propagation), the propagation module might return the empty set when in fact an inconsistency exists. To ensure completeness, we can use some constraint satisfaction procedure (like the one in Qsim), but at a significantly higher computational cost.

Also, the conflict set found by the Qsim constraint-propagation module might not be minimal, since some of the constraints flagged as "used" might not have actually resulted in the contradiction detected. If we are also willing to incur higher computational cost by keeping track of the dependency of the values propagated in terms of the supporting constraints, then we might be able to return more specific conflict sets.

Making Reiter's algorithm incremental. As mentioned earlier, we'd like to be able to perform incremental diagnosis as successive qualitative states of the device are observed. Fortunately, we can modify Reiter's algorithm to perform incremental diagnosis. Figure 3 shows the modified algorithm Inc-Diagnose.

A key observation is that the information expressed by a conflict set is cumulative: Once a conflict set is detected in some state, then at least one of its components must be faulty, even though this might be "hidden" and undetectable in some other state. A conflict set gathered in one state is still applicable across all other states, although new conflict sets might be found as the algorithm examines more states and as more measurements are taken on the device.

An example diagnosis

The qualitative values of all the temperature controller's parameters are shown in Table 1. The observed behavior is a result of two failing components: a temperature

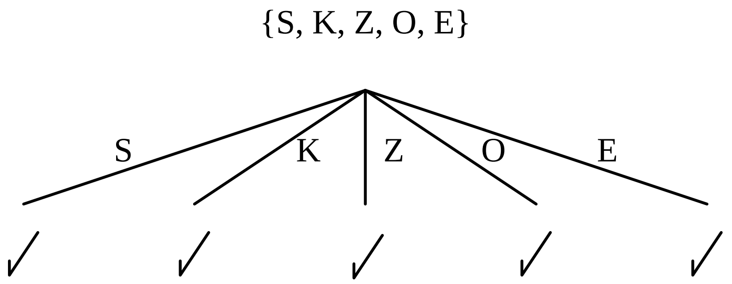

Figure 4: The HS-DAG after observing the room temperature, the temperature set by the control knob, the power, the power switch, and the heat flow at state s_0.

sensor that shows a significant time lag in registering the room temperature, and a malfunctioning proportional controller that is stuck at the zero adjustment level. We assume that initially, we have observations of the values of T_i^{ob}, T_s^{ob}, P^{ob}, W, and HF only, giving rise to three qualitative states s_0, s_1, s_2.

Based on these parameter values, Inc-Diagnose first considers state s_0. For the following discussion regarding sets of components, the temperature sensor is S, the temperature control is K, the proportional controller is Z, the on/off power switch is O, and the electric heating/cooling component is E. Inc-Diagnose calls the Qsim constraint propagator with the set of components $\{S, K, Z, O, E\}$. The constraint propagator returns $\{S, K, Z, O, E\}$ as the conflict set at the root (see Figure 4). Next, it considers the first child node linked to the root by an edge labeled S. It makes another call to the propagator with the components $\{K, Z, O, E\}$. The propagator returns the empty set, and so this child node is completed. Similar calls are made at the other child nodes, and the empty set is returned by the propagator for all of them. Figure 4 shows the HS-DAG after considering state s_0. In all the figures, the completed leaf nodes are indicated by a check mark. The diagnoses at this point are $\{S\}$, $\{K\}$, $\{Z\}$, $\{O\}$, and $\{E\}$.

Next, Inc-Diagnose considers state s_1. It opens all five child nodes and makes successive calls to the propagator with the components $\{K, Z, O, E\}$, $\{S, Z, O, E\}$, $\{S, K, O, E\}$, $\{S, K, Z, E\}$, and $\{S, K, Z, O\}$ on the first, second, ..., and fifth child node respectively. Each of these calls returns the empty set, and so the same HS-DAG remains after processing state s_1. Similar calls are made when processing state s_2, with no changes in the HS-DAG (as shown in Figure 4).

Now suppose we obtain additional measurements of the values of parameter T_i to further differentiate the diagnoses. The values of T_i appear in the sixth row of Table 1. Note that its qualitative value decreases from T_{mid} to Cold and then stabilizes within state s_2. Since the value of T_i undergoes qualitatively distinct changes in state s_2, s_2 splits into three distinct qualitative states: s_2, s_3, s_4. We now have a total of five qualitative states.

Processing state s_0 again with the newly added parameter value, Inc-Diagnose opens all five child nodes. At the first child node, Inc-Diagnose calls the propagator with the components $\{K, Z, O, E\}$. This same set is returned as the conflict set. Using the DAG pruning heuristics described earlier, since $\{K, Z, O, E\}$ is a proper

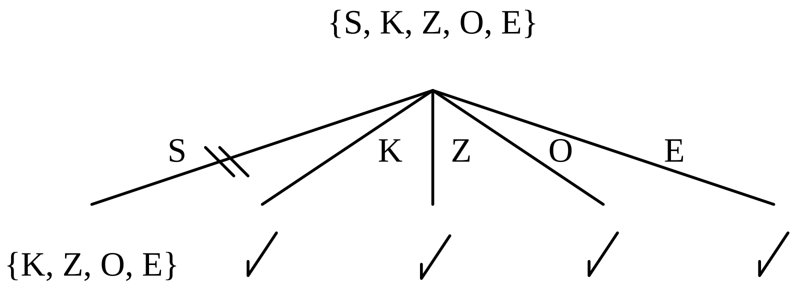

Figure 5: The HS-DAG after observing the temperature recorded by the sensor and processing state s_0.

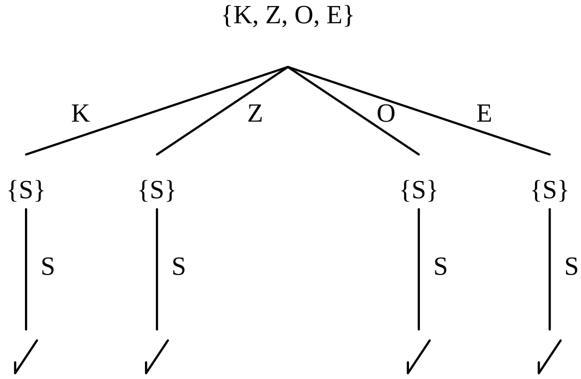

Figure 6: The HS-DAG after observing the temperature recorded by the sensor and processing state s_2.

subset of $\{S, K, Z, O, E\}$ at the root, we can prune away the redundant edge labeled S and its descendants. The calls made at the other child nodes all return empty sets; the resulting HS-DAG after processing state s_0 is shown in Figure 5. At this point, the diagnoses are $\{K\}$, $\{Z\}$, $\{O\}$, and $\{E\}$. Processing the next state s_1 does not yield any changes to the HS-DAG. (Because of the way Qsim checks for corresponding values, the two qualitative magnitudes (Cold,T_{rm}) and (T_{mid},T_{rm}) do not violate the equality constraint, since they are still on the "same side" of both landmark values, Cold and T_{rm}.)

Continuing to process state s_2, Inc-Diagnose calls the propagator with the components $\{S, Z, O, E\}$ at the first child node with the path label $\{K\}$. This returns the conflict set $\{S\}$. At the next child node with path label $\{Z\}$, we can reuse the conflict set $\{S\}$ already found, since $\{S\} \cap \{Z\} = \emptyset$. Similarly $\{S\}$ can be reused for the remaining two child nodes. Proceeding to the next level of the HS-DAG, all calls to the propagator return the empty set. The final HS-DAG is shown in Figure 6. No changes to the HS-DAG occur after processing states s_3 and s_4. Now, the diagnoses are $\{K, S\}$, $\{Z, S\}$, $\{O, S\}$, and $\{E, S\}$.

Next we measure the value of a to be zero and steady. Inc-Diagnose reconsiders state s_0 with this newly added parameter value. Calls to the propagator made at the first two nodes considered with path labels $\{K, S\}$ and $\{Z, S\}$ return the empty

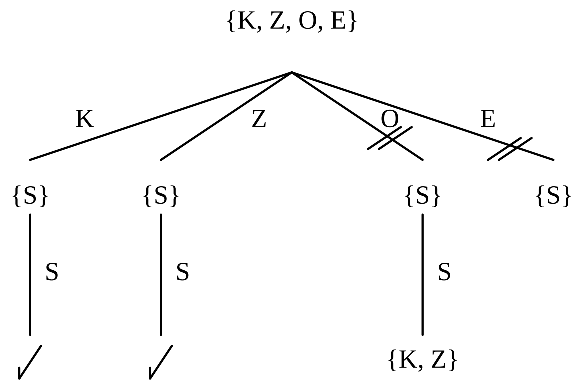

Figure 7: The HS-DAG after observing the adjustment amount and processing state s_0.

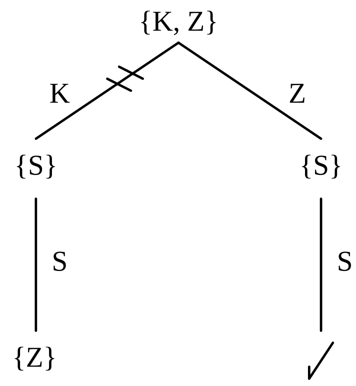

Figure 8: The HS-DAG after observing the temperature set and processing state s_0.

set. The next call at the node with path label $\{O, S\}$ yields the conflict set $\{K, Z\}$. This results in the pruning of the two branches O and E and their descendants from the root, since $\{K, Z\}$ is a proper subset of the conflict set $\{K, Z, O, E\}$ at the root (see Figure 7). The diagnoses are now $\{K, S\}$ and $\{Z, S\}$. No further changes occur when considering $s_1, \ldots, s_4$.

Finally, the measurement T_s is taken. Reconsidering state s_0, the call to the propagator at the node with path label $\{K, S\}$ yields the conflict set $\{Z\}$. The first HS-DAG branch is pruned away, and the second call at the remaining node returns the empty set. Subsequent processing of the other states does not yield any changes. The final diagnosis is $\{Z, S\}$ (see Figure 8). Inc-Diagnose has correctly diagnosed the two faults present in the device.

A prototype system has been successfully implemented and tested on several dynamic devices (a temperature controller, a pressure regulator, and a toaster). The examples took only a few minutes to run on an Explorer/I Lisp machine.

Related work

Other related research in the diagnosis of dynamic continuous systems includes the early work of Pan [7] and Scarl et al. [8], as well as Dvorak and Kuipers' model-based monitoring work [9]. However, these systems are not based on any formal theory of diagnosis, and their diagnosis algorithms are not as general.

As currently implemented, the constraint propagation module does not cache any of its computation. However, separate calls initiated by the Inc-Diagnose algorithm can share much of the same computation. For example, consider the two calls with components $\{S, K, Z, O\}$ and $\{S, K, Z, E\}$. Caching would save significant computation, since the computational work done to propagate through components S, K, and Z is the same in both cases. An assumption-based truth maintenance system [1] could be used to cache partial computation.

Other interesting research issues include the hierarchical decomposition of a device to better manage the computational complexity of computing diagnoses [10], and the use of fault models to better identify the kinds of faults present in a component once they are isolated [9].

Acknowledgments

This research was supported by a Microelectronics and Computer Development Fellowship from the University of Texas at Austin. Thanks to Dan Dvorak, Adam Farquhar, Ben Kuipers, and the anonymous reviewers who provided helpful comments on this article. Special thanks are due to Ben Kuipers and Ray Mooney, who helped greatly in improving the presentation of this work at the Sixth IEEE Conference on AI Applications.

References

[1] J. DeKleer and B. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence*, Vol. 32, No. 1, Apr. 1987, pp. 97–130.

[2] R. Reiter, "A Theory of Diagnosis from First Principles," *Artificial Intelligence*, Vol. 32, No. 1, Apr. 1987, pp. 57–95.

[3] B.J. Kuipers, "Qualitative Simulation," *Artificial Intelligence*, Vol. 29, No. 3, Sept. 1986, pp. 289–338.

[4] B.J. Kuipers, "Commonsense Reasoning about Causality: Deriving Behavior from Structure," *Artificial Intelligence*, Vol. 24, Nos. 1–3, Dec. 1984, pp. 169–203.

[5] R. Greiner, B.A. Smith, and R.W. Wilkerson, "A Correction to the Algorithm in Reiter's Theory of Diagnosis," *Artificial Intelligence*, Vol. 41, No. 1, Nov. 1989, pp. 79–88.

[6] J.A. Reggia, D.S. Nau, and P.Y. Wang, "Diagnostic Expert Systems Based on a Set-Covering Model," *International Journal of Man-Machine Studies*, Vol. 19, No. 5, Nov. 1983, pp. 437–460.

[7] J. Pan, "Qualitative Reasoning with Deep-Level Mechanism Models for Diagnoses of Mechanism Failures," *Proceedings of the First Conference on Artificial Intelligence Applications*, CS Press, Los Alamitos, California, 1984, pp. 295–301.

[8] E. Scarl, J. Jamieson, and C. Delaune, "A Fault Detection and Isolation Method Applied to Liquid Oxygen Loading for the Space Shuttle," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Mateo, California, 1985, pp. 414–416.

[9] D. Dvorak and B. Kuipers, "Model-Based Monitoring of Dynamic Systems," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Mateo, California, 1989, pp. 1238–1243.

[10] W. Hamscher, "XDE: Diagnosing Devices with Hierarchic Structure and Known Component Failure Modes," *Proceedings of the Sixth Conference on Artificial Intelligence Applications*, CS Press, Los Alamitos, California, 1990, pp. 48–54.

Hwee Tou Ng is a doctoral candidate in the Department of Computer Sciences at the University of Texas at Austin. He received his BSc in computer science for data management from the University of Toronto in 1985 and his MS in computer science from Stanford University in 1987. His research interests include artificial intelligence, natural language processing, automated reasoning, and knowledge-based systems. He received the Best Student Paper Award for this work at the Sixth IEEE Conference on AI Applications. He also holds an IBM graduate fellowship.

The author can be reached at the Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712.