A Framework for Model-Based Repair

Ying Sun & Daniel S. Weld*
Department of Computer Science and Engineering, FR-35
University of Washington
Seattle, WA 98195
ysun, weld@cs.washington.edu

Abstract

We describe IRS, a program that combines partial-order planning with GDE-style, model-based diagnosis to achieve an integrated approach to repair. Our system makes three contributions to the field of diagnosis. First, we provide a unified treatment of both information-gathering and state-altering actions via the UWL representation language. Second, we describe a way to use part-replacement operations (in addition to probes) to gather diagnostic information. Finally, we define a cost function for decision making that accounts for both the eventual need to repair broken parts and the dependence of costs on the device state.

Introduction

Although researchers have investigated model-based diagnosis for many years, only recently has attention turned to what should, perhaps, have been the central question all along: repair. When field-replaceable parts contain multiple components, focusing on determining the exact component responsible for faulty behavior can be counterproductive, since the final probes may not distinguish between repair actions. Furthermore, most diagnosis research has assumed that all probes have the same cost, leading to diagnostic strategies guided solely by estimated information gains.

In this paper we argue that both of these problems are best addressed by integrating theories of perception and action. In other words, we claim that repair is best thought of as a marriage of diagnosis and planning. The planner needs to call diagnosis as a subroutine to determine which observations will best improve its incomplete information, and the diagnoser needs to call the planner to estimate the cost of observations that are not directly executable (*e.g.*, probing a location inside a closed cabinet). A rational approach to repair requires accounting for both the cost/benefit tradeoff of actions as well as the synergistic changes in device state that allow one action to facilitate others.

In this paper, we describe an integrated repair system called IRS[1], and discuss three important aspects of its operation.

- Fundamentally, there is no difference between actions that gather information (*i.e.*, probes) and actions that change the device state; they should be treated uniformly. This allows representation of actions that have both state-changing and information-gathering aspects. When estimating the cost of an action that is not directly executable, an agent should add the costs of the primitive actions in a plan that achieves the desired effect.
- The ability to replace parts and repeat observations provides new diagnostic opportunities, similar to those provided by test generation systems. Our integrated theory of action and observation allows a diagnostic agent to combine replacement and observation actions synergistically. A comprehensive utility model selects between strategies.
- When estimating the cost of an operation, it is crucial to consider the eventual cost of repairing broken parts, not just the cost of diagnosis.

The next section of the paper defines an action representation language that distinguishes between observations of and changes to the device state. Then we show how to extend GDE to handle diagnosis of devices with changing state using both traditional observations as well as replacement operations. We decompose our cost function into two parts: the cost of executing the substeps of the current operation, and the expected cost of future diagnosis and repair operations. We show

*This research was funded in part by National Science Foundation Grant IRI-8957302, Office of Naval Research Grant 90-J-1904, and a grant from the Xerox corporation. Our implementation is built on pieces of code that were written in part by Johan de Kleer, Denise Draper, Ken Forbus, Steve Hanks, and Scott Penberthy. In addition to those mentioned above, we benefited from conversations with and comments by Oren Etzioni, Walter Hamscher, Nick Kushmerick, Neal Lesh, Mark Shirley, and Mike Williamson.

[1] IRS stands for Integrated Repair System, but its concern with cost evokes images of another basis for the acronym.

how the UCPOP planner [Penberthy and Weld, 1992] can be used to calculate this cost function, and we illustrate our algorithm on two simple refrigerator [Althouse *et al.*, 1992] examples. After discussing the implementation, we close with a discussion of related and future work.

Modeling Action and Change

The first step in creating a unified theory of repair is to select a generalized model of action that distinguishes between causal and information-gathering effects. For example, it is crucial to differentiate between an *observation* that the voltage of a node is zero and an action that *grounds* the node. Even though the agent knows the voltage is zero in both cases, the effects are very different. Traditional diagnosis systems do only the former, while most implemented planners handle only the latter; an integrated repair system needs both. Even though a whole AI research subfield is devoted to representations of action [McCarthy and Hayes, 1969], most existing theories do not meet our needs:

1. Ability to represent incomplete information.
2. Distinguish between observations (which increase information, but don't change the world state) and actions with causal effects.
3. Computationally tractable.

For example, although the STRIPS representation satisfies the last criterion, it assumes complete information and thus renders the notion of observation meaningless. [Moore, 1985] develops a first-order modal logic that codifies actions that supply an agent with information, and [Morgenstern, 1987] presents a more expressive language that allows actions to have knowledge preconditions, but neither researcher considers algorithms for generating plans using their models of action.

Our UWL representation [Etzioni *et al.*, 1992] is perfectly suited to the needs of repair. An extension of STRIPS that handles incomplete information, UWL was originally designed to represent UNIX commands for a Softbot [Etzioni and Segal, 1992]. The novel aspects of the language include annotations to differentiate causal from observational effects and informational from causational goals, `T`, `F`, and `U` truth values, and run-time variables.

For example, one might write the goal or precondition of setting the voltage of `V` to `220` with `(satisfy (value-of V 220))` while the goal of determining the current voltage at that probe point can be written as `(findout (value-of V ?x))`. Similarly, the effect of an action that grounds a node might be `(cause (value-of V 0))` while a step that just observes the value without changing it would be written as `(observe (value-of V !y))`. In these examples, `?x` denotes a plan-time variable whose value may be constrained during subsequent planning decisions [Stefik, 1981], but `!y` denotes a run-time variable that is treated as an (unknown) constant by the planner and whose value is only established when the plan is executed. Abstractly, a UWL step schemata contains a step name, a set of preconditions, and a set of postconditions (with associated cost). Preconditions and goals are annotated with **satisfy** or **findout**; postconditions are annotated with **cause** and **observe**. Initial conditions are represented with a dummy step that has no preconditions and whose postconditions cause all propositions to take on some truth value, `U` in the case of incomplete information. Complete details and formal semantics are provided in [Etzioni *et al.*, 1992].

To illustrate the use of UWL, we show a simplified version of part of our refrigerator domain theory. When the argument `?x` of a **measure** step is an internal voltage, the probe cost is 5.[2] A **measure** step also causes the proposition `(probed ?x)` to be true and sets `?x` to `!v`, a run-time variable whose value will be determined during plan execution.

```
(define-step (measure ?x)
  :when (and (satisfy (internal-voltage ?x))
             (satisfy (not (backplane-on))))
  :effect (and (cause (probed ?x))
               (observe (value-of ?x !v)))
  :cost 5)
```

Diagnosis with Changing State

As discussed in [Sun and Weld, 1992], a variety of architectures are possible for a repair agent. We choose to put a diagnostic reasoner at the top level with the planner as a subroutine. The diagnosis code maintains a model of the most probable modes of the device's components (candidate sets) and uses the planner to suggest useful action sequences. The most utile action is chosen, the device state and candidate sets are updated, and the process is repeated until the stop criterion is satisfied.

In the remainder of this section, we describe how this planner allows estimation of the costs of different operations in a manner that accounts for both the eventual repair need and the state-dependence. We illustrate our algorithm on two troubleshooting episodes with a domestic refrigerator [Althouse *et al.*, 1992] whose schematic is shown in Figure 1.

Calculating Costs

Suppose that the refrigerator is in state S_1: the refrigerator door is closed, the backplane is attached, the refrigerator is located near the wall, and the power is on; the temperature inside the refrigerator is too warm yet the compressor is not running. In this example, it will turn out that the actual fault is the thermostat (which is stuck open), although IRS, of course, does not know this yet. Assuming that the power supply is

[2] Measuring the compressor status or other `?x` might incur a different cost and have different preconditions.

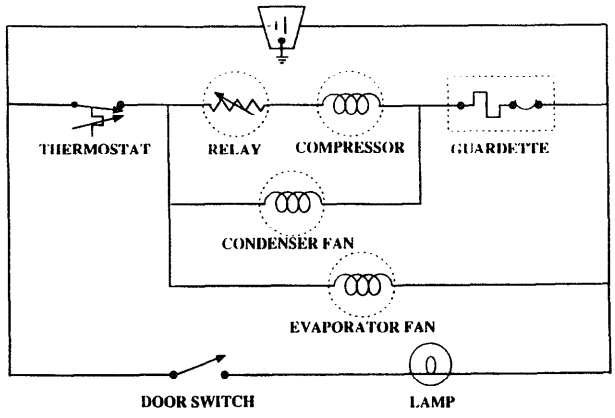

Figure 1: Wiring diagram for a domestic refrigerator

ok and every component has identical prior failure rate (pfr = 0.001), the most probable candidates are:

$$p([\texttt{thermostat1}]) = p([\texttt{relay1}]) = p([\texttt{compressor1}]) = p([\texttt{guardette1}]) \simeq 0.25$$

To find the best operation O_i, various costs must be computed and compared. IRS employs a cost function using **n**-step lookahead:

$$C_{\text{total}}(O_i, \mathbf{S}, \mathbf{n}) = C_{\text{exec}}(\mathcal{P}(\mathbf{S}, O_i)) + \sum_j p(\mathbf{S}_{ij}) EC(\mathbf{S}_{ij}, \mathbf{n}-1) \quad (1)$$

The total cost of executing operation O_i in state **S** as estimated using **n**-step lookahead is equal to the cost of directly executing a plan that achieves the operation plus the weighted sum of the estimated expected costs of the resulting outcomes. $\mathcal{P}$ denotes the planning function that takes an initial state and goal conjunct (encoding a diagnosis or repair operation) as arguments and returns a plan (linearized sequence of primitive actions). Thus, $C_{\text{exec}}(\mathcal{P}(\mathbf{S}, O_i))$ denotes the cost of executing a plan that achieves an operation O_i (*e.g.*, a probe or a replacement) given device state **S**. The expected cost of future operations depends on the outcome of the current operation. For each possible state $\mathbf{S}_{ij}$ resulting from executing the plan for O_i, we compute the expected cost with $(\mathbf{n}-1)$-step lookahead; the cost is then weighted by the probability of each outcome.

The following recursive function computes the expected cost of device state **S** with **n**-step lookahead:

$$EC(\mathbf{S}, \mathbf{n}) = 0 \quad \textit{if } \textit{Reliab}(\mathbf{S}) > 1-\epsilon \quad (2)$$

$$EC(\mathbf{S}, \mathbf{n}) = EC_{\text{repair}}(\mathbf{S}, C) + EC_{\text{diag}}(\mathbf{S}, C) \quad (3)$$

$$= \sum_{c \in C} p(c) C_{\text{exec}}(\mathcal{P}(\mathbf{S}, \Re(c))) + EC_{\text{op}}(C)[-\sum_{c \in C} p(c) log(p(c))] \quad \textit{if } \mathbf{n} = 0$$

$$EC(\mathbf{S}, \mathbf{n}) = \min_{O_i} C_{\text{total}}(O_i, \mathbf{S}, \mathbf{n}) \quad \textit{if } \mathbf{n} > 0 \quad (4)$$

The function *Reliab*(**S**) estimates the reliability of the device in state **S**; repair terminates when the reliability is above the threshold $1-\epsilon$. In the base case, when the lookahead step $\mathbf{n} = 0$, IRS estimates the remaining costs for candidate discrimination and part repair, and sums them. To estimate the repair cost, IRS iterates through the candidates and asks the planner for a plan that replaces all the parts containing a component in that candidate; the cost of that plan is weighted by the probability of the candidate. In the equation above, C denotes the set of candidates in state **S**; $\Re(c)$ denotes the conjunctive goal formula that specifies "Replacement" of all the parts with a component in candidate c. The remaining repair cost also includes the cost of placing any removed but working parts back in the device. The remaining cost for partitioning the candidates is estimated using minimum entropy, where $EC_{\text{op}}(C)$ is the estimated average cost of such an operation (which may expand to multiple actions). When $\mathbf{n} > 0$, IRS estimates the cost to be the minimum cost of the possible operations at each step.

For example, to estimate the cost of probing the status of `condenser-fan1`, IRS calls the planner with the goal `(findout (value-of status-of-cond-fan1 !vcf))` and the initial state of the refrigerator. In this case the planner returns[3] a plan, γ_1, with execution cost $C_{\text{exec}}(\gamma_1) = 4$:

```
(move-refrigerator-away-from-wall)  cost = 2
(measure status-of-condenser-fan1)  cost = 4
```

There are two possible outcomes of probing the status of `condenser-fan1`: with probability 0.5 `!vcf` is `on`, which results in most probable candidates $p([\texttt{relay1}]) = p([\texttt{compressor1}]) \simeq 0.5$; and with probability 0.5 `!vcf` is `off`, which results in most probable candidates $p([\texttt{thermostat1}]) = p([\texttt{guardette1}]) \simeq 0.5$. If 1-step lookahead is used, IRS arrives at the base case at this point.

When `condenser-fan1` is `on`, the estimated repair cost is 56:

```
(disconnect-power)                  cost =  1
(remove-backplane)                  cost = 20
(remove-part relay1/compressor1)    cost =  6
(place-part relay2/compressor2)     cost =  6
(attach-backplane)                  cost = 20
(move-refrigerator-back-to-wall)    cost =  2
(connect-power)                     cost =  1
```

When `condenser-fan1` is `off`, the estimated repair cost is 18 if `thermostat1` turns out to be broken

[3]Space limitations preclude a complete description of our UCPOP partial-order planning algorithm, but it has several desirable attributes: sound, complete, and efficient. The details can be found in [Penberthy and Weld, 1992].

($p \simeq 0.5$) or 56 if `guardette1` turns out to be broken ($p \simeq 0.5$), resulting in an average of 37. In both cases, the estimated cost to discriminate among the remaining candidates is $3.0 * [-(2 * 0.5log0.5)] = 3.0$, where 3.0 is the estimated average cost of such an operation. Therefore, the estimated total cost of diagnosis and repair starting with a probe to the status of `condenser-fan1` is $4 + (0.5 * 56 + 0.5 * 37) + 3.0 = 53.5$.

All other operations cost more at this point, so IRS chooses to probe the status of `condenser-fan1`.

The plan γ_1 is executed, putting the device into state $\mathbf{S}_2$. `condenser-fan1` is observed to be `off`, causing the set of most probable candidates to be updated to:

$p([\texttt{thermostat1}]) = p([\texttt{guardette1}]) \simeq 0.4995$
$p([\texttt{relay1}, \texttt{cond-fan1}]) \simeq 0.0005$
$p([\texttt{compressor1}, \texttt{cond-fan1}]) \simeq 0.0005$

At this point, the costs of all the possible operations are computed again. In addition to considering the option of probing the status of `thermostat1` or `guardette1`, IRS also considers the possibility of *replacing* one of the components. In this case, replacing `thermostat1` happens to be the cheapest operation, with a plan, γ_2, of cost $C_{\text{exec}}(\gamma_2) = 16$ and estimated total cost $C_{\text{total}} = 52$:

```
(disconnect-power)                cost = 1
(open-refrigerator-door)          cost = 1
(remove-part thermostat1)         cost = 6
(place-part  thermostat2)         cost = 6
(close-refrigerator-door)         cost = 1
(connect-power)                   cost = 1
```

Executing this plan leads the device to state $\mathbf{S}_3$.

Computing the costs of all the possible operations reveals that probing the status of `compressor1` (*i.e.*, checking if it is running) has the lowest total cost, so the corresponding plan, γ_3, is executed:

```
(measure status-of-compressor1)  cost = 1
```

The device state is updated to $\mathbf{S}_4$. Observing `compressor1` running exonerates `guardette1` and yields the final candidate:

$p([\texttt{thermostat1}]) \simeq 0.999$

Since `thermostat1` has already been replaced, IRS simply moves the refrigerator back to the original location. At this point, the reliability of the device is 0.999, which is above the preset threshold 0.99, so we are done.

Table 1 summarizes the changing reliability of the device, the possible operations, their costs, and the candidates generated from the executed operations. (If a probe is executed, the value measured is shown after an arrow.) Note how IRS handles the state-dependent probe costs and takes into account the eventual repair cost throughout the diagnosis process.

A Different Example

Interestingly, if we adjust the prior failure rates of the components such that the failure rate of the `guardette` is three times higher than that of the other components, IRS would generate a different sequence of operations. After IRS measures `condenser-fan1` to be `off`, the most probable candidates are $p([\texttt{guardette1}]) \simeq 0.75$ and $p([\texttt{thermostat1}]) \simeq 0.25$. At this point, replacing `thermostat1` is the cheapest operation to execute because there is no need to remove the `backplane`, which is an expensive action. However, IRS realizes that other operations have cheaper total costs when taking into account projected diagnosis and repair operations. Due to the higher failure rate of the `guardette`, the `backplane` will probably need to be opened anyway. Thus, IRS correctly chooses to probe the status of `guardette1` before replacing any components as summarized in Table 2.

Representing Time-Varying State

Due to the inadequacy of the notion of minimal diagnoses, we implemented a diagnosis engine based on the *alibis* principle proposed by [Raiman, 1992]. As a complement to minimal conflicts, minimal alibis specify conditions such as a component must be working if n other components are known to be working. IRS works by incrementally generating minimal alibis, minimal conflicts, and the corresponding set of prime diagnoses.

In addition, we were forced to extend the normal component model to handle devices with changing state. Assumptions such as `ok(relay1)` are unchanged because of the non-intermittency assumption. However, IRS's structural primitives require a temporal component. We distinguish between the role a component plays in a device (*i.e.*, the slot it occupies) and the device instance itself. IRS's system description is written in terms of roles (*e.g.*, the `relay-function`, etc.) A separate set of axioms indicates what instances fill what roles at what times, *e.g.*, `(fills-role relay-function relay1` t_0`)`. The assumptions that distinguish possible worlds involve instances, *e.g.*, `ok(relay1)`, and time tokens.

With these extensions, IRS can reason about swapping out a part, collecting evidence with a replacement part, swapping the original back in, collecting more evidence and so on. As a result, IRS can combine evidence collected at multiple times and involving different sets of component instances.

The IRS implementation has been run on the refrigerator example and several others, including a modified 3-inverter example [Sun and Weld, 1992]. It took approximately 2 minutes to run the refrigerator example on a SUN SPARC.

Related Work

Since IRS's behavior is to choose the operation with the maximum expected utility, it *could* be seen as a

most probable candidates	*relia-bility*	*possible operations*	*exec cost*	*total cost*	*operation executed*
p([thermostat1]) ≃ .25 p([relay1]) ≃ .25 p([compressor1]) ≃ .25 p([guardette1]) ≃ .25	0	probe cond-fan1-status probe relay1-status probe guardette1-status ...	4 27 27	53.5 67.1 67.1	X → off
p([thermostat1]) ≃ .50 p([guardette1]) ≃ .50 ...	0	replace thermostat1 probe guardette1-status replace guardette1 ...	16 25 34	52.0 62.0 70.0	X
p([thermostat1]) ≃ .50 p([guardette1]) ≃ .50 ... pfr(thermostat2)=.001	.495	probe compressor1-status probe cond-fan1-status ...	1 2	37.0 38.0	X → on
p([thermostat1]) ≃ .999	.999 DONE	move refrigerator back	2	2.0	X

Table 1: pfr(all components) = 0.001

most probable candidates	*relia-bility*	*possible operations*	*exec cost*	*total cost*	*operation executed*
p([guardette1]) ≃ .500 p([relay1]) ≃ .167 p([compressor1]) ≃ .167 p([thermostat1]) ≃ .167	0	probe cond-fan1-status probe guardette1-status ...	4 27	56.3 65.7	X → off
p([guardette1]) ≃ .75 p([thermostat1]) ≃ .25 ...	0	probe guardette1-status replace guardette1 replace thermostat1 ...	25 34 16	61.5 63.0 69.0	X → open
p([guardette1]) ≃ 1.0	0	replace guardette1	34	36.0	X
p([guardette1]) ≃ 1.0 pfr(guardette2)=.003	.997 DONE	move refrigerator back	2	2.0	X

Table 2: pfr(guardette) = 3 * pfr(other components)

straightforward application of decision theory to the repair problem. From this perspective, our contribution is a program that *automates* both the identification of alternatives being compared and the cost estimation for those alternatives. In the past this problem (called decision *analysis*) has been left as a task that requires human solution [Howard *et al.*, 1976]. See [Breese *et al.*, 1991] for other work on automating the construction of decision models.

The standard cost evaluation in model-based diagnosis is based on the number of probes needed to distinguish a set of hypotheses. Although [Raiman *et al.*, 1991] and [de Kleer *et al.*, 1991] generalize this notion, both approaches assume fixed probe costs that are specified *a priori*, whereas the costs in our evaluation function are state-dependent. Compared with some work on allowing multiple observation sets and diagnosing devices with changing states [Raiman *et al.*, 1991, Hamscher, 1991, Friedrich and Lackinger, 1991, Ng, 1991], our focus is on extending an intelligent agent to plan for state change rather than having a passive agent diagnose devices with dynamic behavior. Several researchers have attempted to represent system purpose explicitly and integrate repair with diagnosis. For example, [Friedrich *et al.*, 1991] formalizes a repair process with time-dependence, [Poole and Provan, 1991] focuses on the utility and granularity associated with the repair actions, while [McIlraith and Reiter, 1991] discusses how to recognize the relevance of a probe given a goal; but none of these researchers incorporate planning explicitly into their framework. We use planning explicitly to estimate the costs and execute diagnosis and repair operations. We avoid explicit representation of system purpose because repair (replacement) is already intermingled with the diagnosis process. Reconfiguration might be an interesting extension for IRS; we plan to investigate [Crow and Rushby, 1991] more carefully. Planning to minimize breakdown costs is another ability that complements IRS's strengths; it would be straightforward to incorporate [Friedrich *et al.*, 1992]'s time-dependent cost function into our system, but their greedy algorithms are unlikely to extend gracefully to handle the state-dependent probe costs addressed by IRS.

Our research is also similar to work on test generation programs which may also be thought of as a kind of planner that needs to distinguish between controlling and observing the node values in a circuit. Unlike our situation, the goal/subgoal graph for test generation is largely static; this allows predefinition and op-

timization which are impossible in our case, but see [Shirley, 1986, Shirley, 1988].

Conclusion

We have reported on IRS, our preliminary integration of diagnostic and planning algorithms, and argued that it represents progress towards a general theory of repair. Our contributions are three-fold:

- A unified treatment of information-gathering and state-altering actions with the UWL action representation language.
- A method for using part-replacement operations (as well as simple probes) to gather diagnostic information.
- Decision making based on a cost function that takes into account both the eventual cost of repair and the dependence of cost on device state.

In future work, we hope to investigate heuristics for approximating C_{total}, incorporate the cost of computation into the cost function, and integrate UWL's treatment of incomplete information with UCPOP's ability to handle universal quantification.

References

A. D. Althouse, C. H. Turnquist, and A. F. Bracciano. *Modern Refrigeration and Air Conditioning.* The Goodheart-Willcox Company, Inc., 1992.

J. Breese, R. Goldman, and M. Wellman, editors. *Notes from the Ninth National Conference on Artificial Intelligence (AAAI-91) Workshop on Knowledge-Based Construction of Probabilistic and Decision Models.* AAAI, July 1991.

Judith Crow and John Rushby. Model-Based Reconfiguration: Toward an Integration with Diagnosis. In *Proceedings of AAAI-91*, pages 836–841, July 1991.

J. de Kleer, O. Raiman, and M. Shirley. One Step Lookahead is Pretty Good. In *Proceedings of the 2nd International Workshop on Principles of Diagnosis*, October 1991.

Oren Etzioni and Richard Segal. Softbots as testbeds for machine learning. In *Working Notes of the AAAI Spring Symposium on Knowledge Assimilation*, Menlo Park, CA, 1992. AAAI Press.

Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. An Approach to Planning with Incomplete Information. In *Proceedings of KR-92*, October 1992.

G. Friedrich and F. Lackinger. Diagnosing Temporal Misbehavior. In *Proceedings of IJCAI-91*, August 1991.

G. Friedrich, G. Gottlob, and W. Nejdl. Formalizing the Repair Process. In *Proceedings of the 2nd International Workshop on Principles of Diagnosis*, October 1991.

G. Friedrich, , and W. Nejdl. Choosing Observations and Actions in Model Based Diagnosis / Repair Systems. In *Proceedings of KR-92*, October 1992.

W.C. Hamscher. Modeling Digital Circuits for Troubleshooting. *Artificial Intelligence*, 51(1–3):223–272, October 1991.

R. Howard, J. Matheson, and K. Miller. *Readings in decision analysis.* Stanford Research Institute, Menlo Park, CA, 1976.

J. McCarthy and P. J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

S. McIlraith and R. Reiter. On Experiments for Hypotherical Reasoning. In *Proceedings of the 2nd International Workshop on Principles of Diagnosis*, October 1991.

R.C. Moore. A Formal Theory of Knowledge and Action. In *Formal Theories of the Commonsense World.* Ablex, 1985.

Leora Morgenstern. Knowledge preconditions for actions and plans. In *Proceedings of IJCAI-87*, 1987.

H.T. Ng. Model-based, Multiple Fault Diagnosis of Dynamic, Continuous Physical Devices. *IEEE Expert*, December 1991.

J.S. Penberthy and D. Weld. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In *Proceedings of KR-92*, pages 103–114, October 1992.

D. Poole and G. Provan. Use and Granularity in Consistent-Based Diagnosis. In *Proceedings of the 2nd International Workshop on Principles of Diagnosis*, October 1991.

O. Raiman, J. de Kleer, V. Saraswat, and M. Shirley. Characterizing Non-intermittent Faults. In *Proceedings of AAAI-91*, July 1991.

O. Raiman. *The Alibi Principle*, pages 66–70. Morgan Kaufmann, 1992.

M. Shirley. Generating Tests by Exploiting Designed Behavior. In *Proceedings AAAI-86*, pages 884–890, August 1986.

M. Shirley. Generating Circuit Tests by Exploiting Designed Behavior. AI-TR-1099, MIT AI Lab, December 1988.

M. Stefik. Planning with Constraints (MOLGEN: Part 1). *Artificial Intelligence*, 14(2), 1981.

Y. Sun and D. Weld. Beyond Simple Observation: Planning to Diagnose. In *Proceedings of the 3rd International Workshop on Principles of Diagnosis*, pages 67–75, October 1992.