Controlling inequality reasoning in a TMS-based analog diagnosis system

David Jerald Goldstone*
MIT A.I. Laboratory and Xerox PARC
155 Chestnut St.
Montclair, NJ 07042
stone@ai.mit.edu

Abstract

A system, called *Skordos*, has been implemented for model-based diagnosis of analog circuits. One of the difficulties of model-based diagnosis for analog circuits is managing the tremendous number of predictions which may be generated by a constraint propagation system. Fortunately, not all of those predictions are valuable for the diagnostic process. A process called *hibernation*, which is used in *Skordos* to prevent generation of useless predictions, is introduced and described here. Another technique is introduced and described whcih further assists in controlling the inequality reasoning by exploiting hibernation. This technique involves changing the structure in which values are combined. It uses hibernation as an early filter to reduce the number of interactions resulting from Kirchhoff's current law from exponential to quadratic in the number of interacting variables.

Introduction

Skordos (Goldstone 1991) is a model-based diagnosis system for analog circuits. It is designed in the fashion of *GDE* (de Kleer and Williams 1987): it uses a constraint propagation system with a truth maintenance system (de Kleer 1986). In the analog domain, however, the values of state variables are not known exactly, so the constraint propagation system makes predictions of the form $v > 3.2$, rather than $v = 4$. It turns out that the constraint propagation system generates an overwhelming number of predictions for each state variable. In cases where predictions consist of assignments of values to variables, two distinct predictions for the same variable result in a contradiction. However, when predictions involve inequalities, two predictions for the same variable can support each other. For example, $v < 3.2$ supports $v < 4.4$.

*This paper is based on a thesis submitted on January 22, 1991 to the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology in partial fulfillment of the requirements for the degrees of Master of Science and Bachelor of Science.

Various levels of precision for the same state variable result from propagating constraints through different paths in the circuit. Although multiple values for a variable are allowed, it is not obvious that each must be maintained. Why not keep track of only the strongest inequality? For example, given $v < 5$ and $v < 6$, why not ignore the latter, less precise prediction? The diagnostician must guarantee to keep track of each predicted value for a variable which promises to contribute even the slightest bit of additional information. In addition, every prediction is accompanied by sets of assumptions under which it holds true: $v < 5$ may depend on the correctness of component A; $v < 6$ may depend on the correctness of component B. The less precise prediction is important, for if A were discovered to be faulty, then the less precise prediction remains, whereas the more precise prediction is no longer valid. More generally:

- Given two predictions, Ψ_1 and Ψ_2, where the inequality associated with Ψ_1 is more precise than the inequality associated with Ψ_2, Ψ_2 must not be ignored, in case the set of assumptions associated with Ψ_1 turns out to be unsound.

Multiple sets of predictions must then be propagated around the system, at computational expense.

When many state variables interact, as in n terminals converging which must obey Kirchhoff's current law, the management problem is compounded. As an illustration, consider the case that m predictions have been made for each terminal. Although it may be tempting to combine only a few of the strictest values, they are frequently based on suspect assumptions; more valuable information can come from combinations of less strict predictions. For each of the terminals, some combination of the values at the other $n-1$ terminals might provide useful new information. Trying all the combinations takes m^{n-1} computations. For all the n terminals, $n \times m^{n-1}$ new computations are necessary.

However, by use of a process called *hibernation*, the less precise predictions are prevented from generating more predictions. They are 'deactivated' until the more precise predictions come under suspicion, and

then they are reactivated, as needed. Hibernation will be described in more detail, and that description will show that hibernation does not lose any diagnostic information. Hibernation simply takes advantage of diagnostic conditions which allow the troubleshooter to diminish the predictions per value which must be maintained.

Another technique is used to deal with the explosive number of computations which result from interactions among the remaining active predictions for the variables. This is shown later in the paper. It basically involves restructuring the problem to exploit filters in the hibernation process.

First, a few definitions:

- Σ is used to denote a set of assumptions, where each assumption describes a component in a mode of operation. A is used to denote the assumption that component 'A' is in a normal mode of operation.
- Φ is used to denote an inequality between a state variable and a number, of the form $v > 5$.
- A prediction, Ψ, consists of an inequality and the sets of assumptions on which it is based, of the form $(\Sigma_a, \Sigma_b) — \Phi$. In the example above, one prediction is $v < 5$ $((A))$. If it were shown, by another path, that $v < 5$ assuming some C were working, then the prediction would have two sets of assumptions: $((A),(C))$.
- A prediction, Ψ_1, *subsumes* another prediction, Ψ_2, if the inequality associated with Ψ_1 is stronger than the one associated with Ψ_2 mathematically and if every set of assumptions associated with Ψ_2 is subsumed logically by some set of assumptions associated with Ψ_1. *output*< 16.5 $((C))$ *subsumes output*< 17.5 $((A,C))$
- *Hibernation* is the process by which selected predictions either invoke or are disabled from invoking the constraint propagation system of which they are a part. The selection is based on attempts to decide whether invoking the constraint propagation system will probably generate no new diagnostic information.
- When a prediction is *active*, it is used normally by the constraint propagation system.
- When a prediction is *hibernated*, it does not lead to any new predictions. The constraint propagation system stores up the constraints which operate on it, in case it is ever activated.

Overview of the hibernation process

As introduced above, hibernation is a process by which some less precise predictions are stored, and are not used to generate other predictions. These predictions are chosen as the ones which which provably will not generate new useful diagnostic information.

But what is useful information? And how may it be decided properly whether a prediction will provide that useful information? The diagnostic process centers around *symptoms*, which are differences between observed values and predicted values. Within the framework of *GDE*(de Kleer and Williams, 1987), those symptoms are used to isolate *conflicts*, which are sets of component behavioral modes which are inconsistent with the system description and some observation. A prediction is considered to provide useful information if it might lead to a conflict to which no other active prediction would lead. Therefore, in order to hibernate a prediction, *Skordos* must prove that, at least for the time being, that prediction will generate no new such conflicts.

A straightforward circumstance which leads to hibernation is the subsumption of a prediction. If *subsumes*(Ψ_1, Ψ_2), then any minimal conflict to which Ψ_1 would lead, Ψ_2 would also lead. Thus, upon discovery of both, Ψ_2 may be properly put into hibernation. Hibernation in the case of subsumption can help ease the strains of managing many levels of precision.

In fact, this simple sort of hibernation is needed just to get the system to work. For example, consider a resistor, R_1, with $100 < r_1 < 110$ where it has been observed that $11 < v_1 < 12$. Call this observation Ψ_1. The constraint propagation system infers that $0.1 < i < 0.12$ $[(R_1)]$, and then goes back to infer that $10 < r_1 < 13.2$ $[(R_1)]$. Call this prediction Ψ_2. If this process continued, then the propagator would go into an infinite loop. However, Ψ_2 clearly provides no new information — it is subsumed by Ψ_1. Therefore, Ψ_1 is hibernated. Such subsumptive situations fit nicely into the hibernation framework.

Unfortunately, subsumptive hibernation is not nearly enough to stem the tide of information spewing from the predictive engine. More generally, hibernation may be used with many predictions which are not subsumed. These predictions must be proven not to lead to a conflict, typically by comparison with stronger active predictions which do not lead to a conflict.

For example, if a prediction like $v > 5.95$ does not lead to a conflict, then a prediction which it is mathematically stronger than, like $v > 5$, cannot lead to a conflict either. As another example, suppose Ψ_1 consists of the inequality $v > 5.95$ and the single assumption set (A, B); Ψ_2 consists of $v > 5$ and the assumption set (A); and Ψ_3 consists of $v > 1$ and (B). If there are no conflicts, then Ψ_1 should be the only one active. Considering it is mathematically stronger than the other predictions, and it, in particular, did not lead to a conflict, those weaker predictions will not lead to conflicts either.

Now suppose (A, C) is shown to be the single minimal conflict. In this case, Ψ_1 might have led to the conflict, because it depended on the possibly faulty A. If one of the hibernated predictions might lead to a new

conflict, it should be activated. It might seem that because B, another member of Ψ_1's assumption set, was not implicated in the conflict, Ψ_1 could not have contributed to the conflict. Consider the case that two conflicts had arisen: (A, B, C) (because of this prediction) and (A, C), then the minimal conflict would be (A, C). Therefore, the hibernated predictions must be considered for activation.

Ψ_2, the next strongest prediction mathematically, should not be activated, because it depends on A, which was the only component on which $v > 5.95$ depended which was implicated. It would only come up with redundant conflicts, if any. Ψ_3, although it is much weaker mathematically, should be activated, because it is dependent only on B, on which Ψ_1 is not dependent, and would provide useful information if it were involved in a conflict.

A careful description

In order to maintain correctness of the diagnostic process, a new prediction, Ψ_i, may be put into hibernation only when it is guaranteed to create no new conflicts. Similarly, after having been hibernated for some time, if the guarantee no longer holds, the prediction must be activated. The emphasis is on the creation of conflicts, because the diagnostic process used in *Skordos* bases its reasoning on conflicts. If no new conflicts are created, then no useful information has been learned.

Conflicts are considered to be part of the state of the system. A conflict is of the form $\Gamma_k \rightarrow F$, where Γ_k denotes a set of assumptions and F denotes false. Γ is used, rather than Σ to easily differentiate conflicting sets of assumptions from non-conflicting ones.

Under what conditions may *Skordos* conclude that it will generate no new conflicts from a certain new prediction, Ψ_*? Certainly, for the set of predictions where Φ_* and Φ_i are bounds on the same variable and Φ_i implies Φ_*, then if for every Σ_{*a}, there is some Ψ_j with some Σ_{ja} implies Σ_{*a}, then Ψ_* may be hibernated.

Furthermore for every Σ_{*a}, a subtle weaker condition will suffice: given the same set of predictions mathematically stronger than Ψ_*, that there is some Γ_k and some Σ_{ja} such that $\Gamma_k \bigcap \Sigma_{ja}$ implies Σ_{*a}, then Ψ_* may be hibernated.

A condition which is not any weaker but suggests an efficient implementation follows: For every Γ_k, for every Σ_{*a}, if there exists some activated prediction, Ψ_i, whose inequality is mathematically stronger than Ψ_*'s, and $\Gamma_k \bigcap \Sigma_{ia}$ implies Σ_{*a}, then Ψ_* may be hibernated.

The invariant which implements this behavior is:

- For each Γ_k, the Ψ_i with the strongest inequality which has some Σ_{ia} such that Γ_k does not imply Σ_{ia} should be activated.

In extending these techniques to diagnostic systems which consider faulty modes of behavior, it can be useful to make use of the term diagnosis, with the following definition:

- A *diagnosis* is an assignment of modes (good or faulty), consistent with the given observations, for all the components in the system (de Kleer and Williams 1989).

Because they describe assumptions which are *consistent* with the observations, diagnoses bear an inverse relationship to conflicts. A prediction Ψ_i is *valid* under a diagnosis if there is some Σ_{ia} such that the diagnosis implies Σ_{ia}.

In this terminology, the invariant which implements this behavior is:

- For each diagnosis, the valid Ψ_i with the strongest inequality should be activated.

This invariant will not necessarily lead to maximal hibernation, i.e. hibernating all predictions which won't lead to conflicts. Maximal hibernation requires an omniscience of knowing in advance which prediction would or would not lead to a conflict. Without such omniscience, we resort to implementing the invariant above. This condition will allow hibernation of many predictions. It is guaranteed not to over-hibernate: if a stronger Ψ_i were valid, then because it did not lead to any conflict, a weaker Ψ_* certainly won't lead to any conflict. This invariant does put an upper bound on the number of active predictions: the number of valid diagnoses.

That invariant can be used to generate an algorithm for deciding whether a new prediction should be hibernated, denoted HIB?(Ψ_*):

1. When a new prediction Ψ_* arises for a node in the circuit,
2. For each diagnosis under which Ψ_* is valid,
3. For each active prediction Ψ_i for that node which is valid under that diagnosis,
4. If any Ψ_i is mathematically stronger than Ψ_* then go on to the next diagnosis. (It does not matter which prediction; even if that one itself is hibernated later, it will be hibernated with respect to some other prediction which is even stronger for every diagnosis. Because mathematical strictness is transitive, that prediction will also be stronger than Ψ_*.)
5. If none of them is stronger than Ψ_*, then HIB?(Ψ_*) is FALSE.
6. If all the diagnoses are exhausted without HIB?(Ψ_*) being FALSE, then HIB?(Ψ_*) is TRUE.

If HIB?(Ψ_*) is FALSE, then Ψ_* is added to the active predictions as Ψ_{m+1}. However, it might cause the hibernation of some of the previously active predictions. For each of $\Psi_{1 \ldots m}$, HIB? must be checked. Actually, only the predictions which Ψ_{m+1} is not stronger than must be checked for going into hibernation. If HIB?(Ψ_*) is true, then Ψ_* is added to the hibernated predictions.

The basic invariant of the system is that for every diagnosis, the strongest valid prediction at a node in

the circuit is active. However, when a new conflict arises, the set of diagnoses change. Consequently, the active predictions might deserve hibernation, and the hibernated prediction might need activation. The algorithm for recomputing the hibernating set is somewhat expensive:

1. At each value,
2. For each of the new diagnoses,
3. Go through all the predictions, hibernated and active, and find the strongest one which holds. It can be helpful to keep the predictions sorted by strength for this step.
4. When done with all the new diagnoses at a value, go through each of the active predictions and check HIB? on them.

By using the algorithms above, hibernation may be executed in order to help keep the number of predictions manageable without losing any useful information.

Exploiting the hibernation

In spite of the gains which hibernation allows, *Skordos* must manage multiple predictions for many values. The computation is exponential in the number of interacting variables, as described in the introduction. When many values interact, e.g. computations associated with Kirchhoff's current law, the actual costs become prohibitive. As it turns out, these combinations are often fruitless, or their results frequently are subsumed and hibernate immediately. Executing the exponential number of possible combinations, when only a few ever will prove useful, is wasteful. By restructuring the problem, to use hibernation to enforce early pruning the interactions can be reduced from exponential to quadratic. Methods for restructuring this problem and an analysis of the resources used are presented in this section.

Because it a central example to much of the discussion in this section, the reader should be reminded of Kirchhoff's current law. The algebraic sum of currents entering any node must be zero. In terms of *Skordos*, this law is realized by summing $n-1$ predictions for values of terminals to a circuit node, and concluding with a prediction of the value for the last terminal equal to the opposite of the sum. For example, in figure 1, the predictions at converging terminals A, B and C allow a conclusion for the terminal D. Namely, if at least one milli-ampere of current were entering each of A, B, and C, then at least three milli-amperes must be leaving through D. (The convention is that current is assumed to be entering the circuit node through the terminal, so the conclusion is written as $i_D < -3mA$.)

Such predictions may be made for any terminal of a circuit node, given predictions for all the other terminals. However, such a situation is computationally

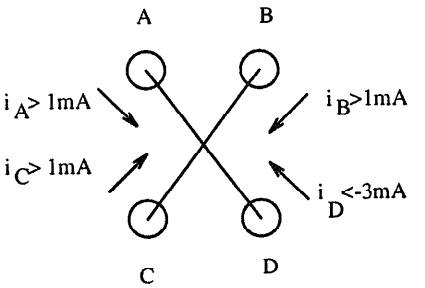

Figure 1: Kirchhoff's current law at a node

explosive, because it involves combining every combination of one prediction from each terminal. With many active predictions and many terminals, the situation borders on disaster. In order to guarantee correctness every combination of inequalities should be tried. However, many of those combinations simply yield useless information.

This problem is exacerbated in computations with many inputs. For example, consider the case of n terminals converging on a circuit node, and m active predictions per value. In applying Kirchhoff's current law, m^{n-1} computations will be required in order to make a complete set of predictions about the first terminal. Because this computation occurs for each terminal, the total cost of computation is $n \times m^{n-1}$. All but about m will be hibernated. Of course, a few of the computations will result in useful information, which will cause a few more computations, until the system reaches stability. Still, the cost of the computation is approximately $O(n \times m^n)$. For a typical circuit node of $n = 6$ inputs and $m = 6$ active predictions per terminal, the total number of computations would be 46,656.

One solution to this problem is used in *Skordos*. It restricts the combinatorics which come with exponential problems of a high degree to a degree of two.

The problem basically requires analysis of the basis of the low yield of active predictions. Typically, predictions from two of the terminals will combine to produce a useless intermediate prediction, which must be combined with all combinations of predictions from all the other terminals. The resulting prediction will be useless as well, in all cases. Consequently, the resulting prediction will be hibernated.

This process, which uses hibernation as a final filter, filters the predictions too late. It is frequently clear, when two active predictions from two values are combined, that the combination is not useful. The solution, then, is to use hibernation as an early filter as well as a final filter. The intuition is to create a new system value, which does not correspond to a particular node in the circuit, but corresponds to a symbolic combination of a subset of the values which are combined

at a circuit node. The intermediate combinations are retained at the new system value explicitly. If a new one is discovered, it is hibernated or activated, in the normal fashion. Because hibernation acts as such a powerful filter on prediction sets, it severely cuts down on the combinatorics of the system. Also, the exponential explosion is limited because the number of combinations depends directly on the number of activated values, and the values are activated in sets whose size is bounded above by the number of diagnoses under analysis.

For example, in Figure 2, the network on the left would be converted, with an intermediate value, to the network on the right. For figure 2, i_m will denote the current into the circuit through `term-m` and i_5 will denote the current from left to right through `inter-term-5`. In order to use Kirchhoff's current law to compute i_3, the straightforward structure of the circuit on the left would try every combination of active predictions for i_1, i_2 and i_4. However, in the circuit on the right, the filtered results of combining i_2 and i_4 would be in i_5. Consequently, predictions for i_3 would be a combination of predictions of i_4 and i_5. By restructuring the interactions, the equations are revised as follows:

$$\begin{aligned} i_1 + i_2 + i_3 + i_4 &= 0 \\ &\Downarrow \\ i_1 + i_3 - i_5 &= 0 \\ i_2 + i_4 + i_5 &= 0 \end{aligned}$$

Loosely, if each terminal had m active predictions associated with it, then the calculation of predictions for `term-3` on the left would take m^3 computations, as contrasted with $2m^2$ on the right. In fairness, i_3 also probably started off with m active predictions, which probably lead to another $2m^2$ computations. Reasoning about these systems is difficult, because computations will create a few new active predictions, which lead to more computations. However, the important lesson is that the order of growth of the computations was reduced, in this simple case, from cubic to quadratic.

The basis for the problem itself provides the intuition that this sort of restructuring maintains the integrity of the system. These terminals represent terminals in an analog electrical circuit which connect in a node. An electrical node represents an equi-potential; it is simply a set of terminals connected by wires. But these wires do not actually connect at a point: they connect in pairs, fusing their currents and reaching an equilibrium with the other terminals. This restructuring simply is a mirror of not joining at a single point.

For the restructuring to take place, $n-3$ new intermediate nodes must be introduced and arranged in a network such that nodes converge in groups of three. In figure 3, for example, each intermediate value progressively represents the sum of one more terminal value.

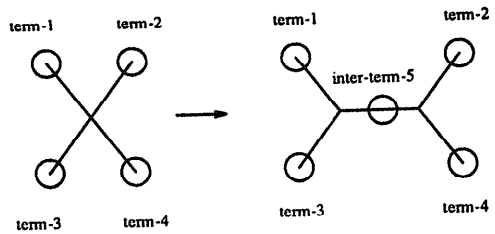

Figure 2: Restructuring the interactions of Kirchhoff's current law

For figure 3, i_m will denote the current into the circuit through `term-m` and, i_n will denote the current directed down through `inter-term-n`. Each intermediate value is summed with the next input to create the next intermediate value, until all the inputs are connected, with the end conditions handled appropriately.

The restructuring of the equations in this case is:

$$\begin{aligned} i_1 + i_2 + i_3 + i_4 + i_5 + i_6 &= 0 \\ &\Downarrow \\ i_1 + i_4 - i_7 &= 0 \\ i_5 - i_8 + i_7 &= 0 \\ i_9 - i_2 + i_8 &= 0 \\ i_3 - i_6 + i_9 &= 0 \end{aligned}$$

Because each node has three terminals, m^2 computations are needed to compute the predictions for each terminal. Therefore, the computation at each node takes $3 * m^2$ time. There are $n-2$ of these nodes, and the whole computation takes about $(n-2)3 * m^2$ time, which is $O(n * m^2)$, which represents a substantial improvement. For a typical circuit node of $n = 6$ terminals and $m = 6$ predictions, about 432 computations are necessary, a savings of a factor of one-hundred over the original problem.

Related Work

The *Sophie* project (Brown, Burton and de Kleer 1982) includes a stand-alone circuit diagnoser which was used as a starting point for this work. *Sophie* greatly exploits the assumption that only one component in the system is failing at a time. *Sophie* included simple versions of many of the ideas which were further developed in *Skordos* in a more recent diagnostic framework (de Kleer and Williams 1987). Another model-based system for diagnosis of analog circuits is called *Dedale* (Dague, Raiman, and Deves 1987). Diagnosis is based on "the fundamental assumption that a defect in a component leads to significant changes in the behavior of the circuit." For instance, to describe

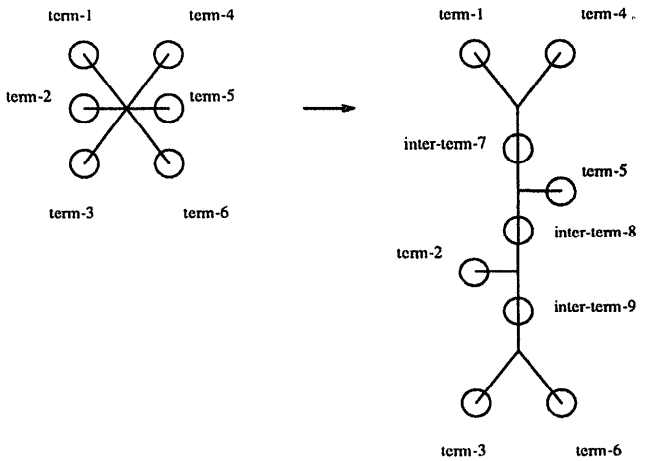

Figure 3: Restructuring with six terminals

state variables. *Dedale* employs an order of magnitude reasoning system, rather than inequalities strictly between state variables and numbers, as described here. In current work in diagnosis of analog circuits, the idea of hibernation is related to the interval refinement techniques described in (Dague, Jehl, and Taillibert 1990) and (Dague et al., 1990.)

Skordos benefitted directly from the work described in (de Kleer 1991) for the framework of the underlying reasoning engine.

Summary

Two methods have been described which help control the inequality reasoning for diagnosis of analog systems. Hibernation is used to deactivate predictions which provably won't lead to any useful information. Because, at most, only one prediction for each value per diagnosis is necessarily active, and the set of diagnoses tends to be small compared to the number of predictions at each value, hibernation can lead to great computational savings. By restructuring interactions which involve many values, hibernation may be used as an early filter to prevent the generation of many useless predictions. This technique reduces the computational complexity of such problems from exponential to quadratic in the number of active predictions per node. Of course, these improvements depend on typical patterns of hibernation: worst case scenarios could conceivably lead to no gains from using hibernation. In practice these techniques are essential in keeping the interactions among the predictions of the constraint propagation system manageable.

The second half of this paper centers on the role of Kirchoff's current law and high direct connectivity both as the cause of bookkeeping difficulties and as the focus to restructure the problem. The dual, Kirchhoff's voltage law and the high number of loops which may be used for reasoning in analog circuits, lead to similar difficulties and may be similarly improved.

Acknowledgments

The author would like to thank Johan de Kleer, Randall Davis and Brian Williams for their guidance with this work. The author would like to thank the reviewers for their valuable comments. Facitilities at the MIT A.I. Laboratory and Xerox PARC were used to study the issues described here.

References

John Seely Brown, Richard Burton, and Johan de Kleer 1982. Pedagogical, natural language and knowledge engineering techniques in Sophie I, II, and III. In D. Sleeman and J. S. Brown, editors, *Intelligent Tutoring Systems*, pages 227–282. Academic Press, New York.

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

Johan de Kleer 1991. Beyond minimal diagnoses. In Proceedings of the National Conference on Artificial Intelligence, Anaheim, California: American Association for Artificial Intelligence, Inc.

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

Johan de Kleer and Brian C. Williams 1989. Diagnosis with behavioral modes. In Proceedings of the Eleventh Joint Conference on Artificial Intelligence, 1324–1330. Detroit, Michigan: International Joint Conferences on Artificial Intelligence, Inc.

P. Dague, P. Deves, P. Luciani, and P. Taillibert, 1990. Analog Systems Diagnosis. In Proceedings of the European Conference on Artificial Intelligence. Stockholm, Sweden.

Philippe Dague, Olivier Jehl and Patrick Taillibert, 1990. An interval propagation and conflict recognition engine for diagnosing continuous dynamic systems. In *Expert Systems in Engineering*, pages 16–31, Springer-Verlag.

Philippe Dague, Oliver Raiman, and Philippe Deves 1987. Troubleshooting: when modeling is the trouble. In Proceedings of the Tenth Joint Conference on Artificial Intelligence, 600–605. Milan, Italy: International Joint Conferences on Artificial Intelligence, Inc.

David Jerald Goldstone 1991. *Applying Model-based Diagnostic Techniques to Analog Circuits* M.S. thesis, MIT.