SPEEDING UP MODEL-BASED DIAGNOSIS BY A HEURISTIC APPROACH TO SOLVING SAT

Benno Stein
Faculty of Media / Media Systems
Bauhaus University Weimar, Germany
benno.stein@medien.uni-weimar.de

Oliver Niggemann
dSPACE GmBH, Germany
ONiggemann@dspace.de

Theodor Lettmann
Computer Science Dept
University of Paderborn
lettmann@upb.de

ABSTRACT
Model-based diagnosis of technical systems requires both a simulation machinery and a logic calculus. The former is responsible for the system's behavior analysis, the latter controls the actual diagnosis process. Especially when pursuing qualitative simulation, it makes sense to realize the simulation machinery with a logic calculus as well. Say, a qualitatively described hypothesis can directly be mapped onto an instance of the well-known SAT problem. Likewise, an entire diagnosis process, i. e., a sequence of hypothesis refinements, represents a set of SAT problems.

This paper reports on the operationalization of such a SAT-based diagnosis approach. A specific characteristic here is the idea to exploit an ordering of the logical formulas according to their likeliness of being satisfiable. This idea is new in the context of qualitative reasoning, and it leads to a considerable speed up of the diagnosis process. Its applicability has been evaluated in the domain of hydraulic circuit diagnosis.

KEY WORDS
diagnosis, machine learning, model-based reasoning, qualitative reasoning, SAT problem

1 Model-Based Diagnosis, Qualitative Modeling, and Satisfiability

This section introduces the idea of model-based diagnosis and a special qualitative modeling approach for modular technical systems. So far, our modeling approach has only be used in the domain of fluidic engineering; however, it is not tailored to a particular plant structure but allows for the generation of behavior descriptions for a large class of circuits.

The qualitative modeling happens within two steps: A precise numerical behavior analysis, which in turn is used to generate a compact qualitative behavior description. In this way, the large analysis search space, which is a common problem when qualitatively simulating fluidic or electrical systems whose behavior is grounded on flow and potential constraints, is kept minimum.

The qualitative simulation process is coded as a satisfiability (SAT) problem within propositional logic. Recall that in the course of diagnosing a system a lot of simulation runs may take place. This corresponds to the problem of identifying satisfiable formulas within a set Ψ of formulas, where each element $\psi \in \Psi$ encodes a single diagnosis hypothesis.

The paper in hand focuses onto this situation. It shows that it is possible to learn an analysis order for diagnosis environments Ψ such that significantly less formulas have to be analyzed—say, SAT problems have to be solved—to obtain a fixed number of satisfiable formulas $\psi \in \Psi$. This paper will not engage into details with respect to qualitative modeling or model-based diagnosis but outline the employed ideas. The activity diagram in Figure 1 gives an overview.

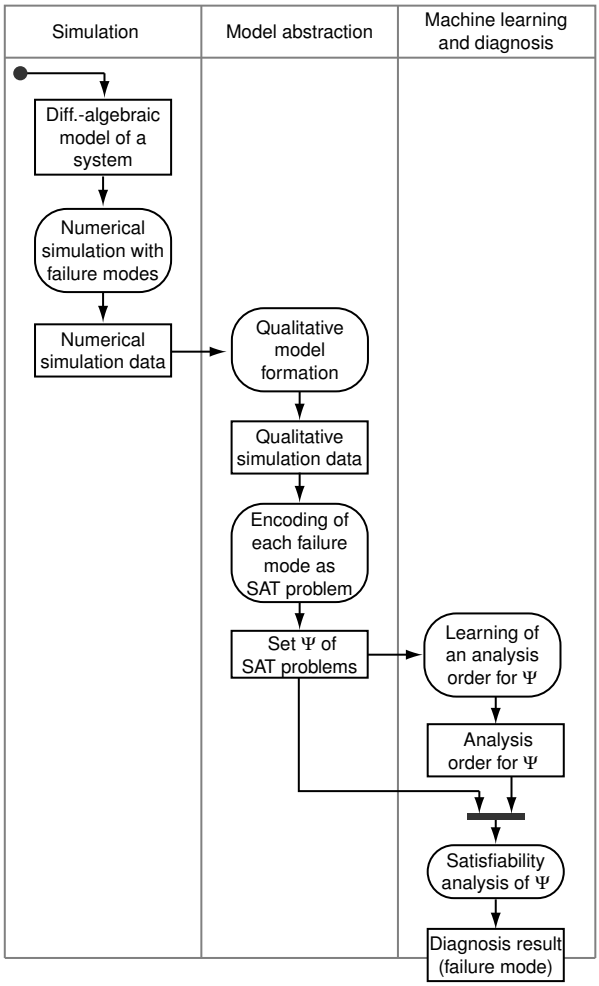

Figure 1. Activity diagram of the diagnosis approach.

Model-based diagnosis approaches employ a "deep model" of the domain and the system under investigation. By modeling functional dependencies in the form of physical cause-effect-relations, the behavior of the interesting system is simulated at some level of accuracy. Typically, the behavior of the entire system is a result of the interplay of local behavior descriptions of the system's components.

During the diagnosis process, the simulated behavior of the model is compared to the observed behavior of the faulty system. Objective is to explain, say, to match both simulated and observed behavior. In this connection several approaches have been developed, such as the GDE,[1] GDE+, Sherlock, or Diagnosis from First Principles [8, 14, 5, 6, 18, 4, 17]. These approaches can be distinguished by their ability to model fault behavior,[2] by the integration of methods from the field of statistics or information theory, or by the strategy a user is guided when comparing the real system to the simulated model.

Common to all model-based diagnoses approaches is the concept of a *conflict*, which defines a set of components that cannot behave correctly at the same time—if the observed behavior shall be explained by the interplay of all components. Hence, at least one defect component must be among each conflict. Given a misbehaving system S, the determination of the minimum conflict sets is a central and, perhaps, the most difficult job since it requires multiple simulations of S.

1.1 A Qualitative Model for Hydraulic Circuit Diagnosis

Hydraulic circuits consist of cylinders that transform hydraulic energy into mechanical energy, various forms of valves, which control flow and pressure of the hydraulic medium, and service components such as pumps, tanks, and pipes, which provide and distribute the necessary pressure p and flow Q. Figure 2 shows an example.

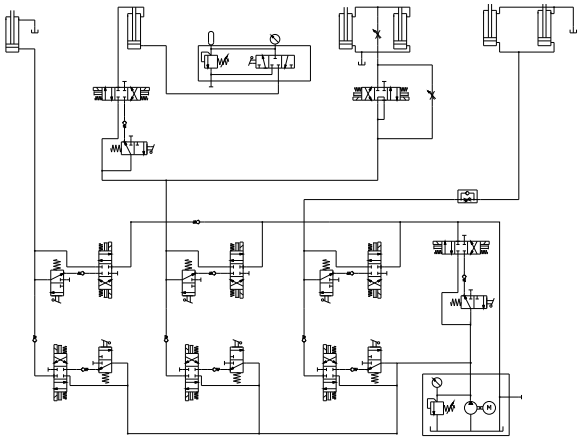

Figure 2. Example of a hydraulic circuit.

[1] GDE stands for "general diagnosis engine".

[2] Fault behavior may be modeled explicitly, by means of fault models, or implicitly, by the absence of the component's intended O.K.-behavior.

Like other technical systems hydraulic circuits can break down. Given this case typical symptoms are observed at the cylinder, whose piston may extend too slowly or may drift. The cause for such a misbehavior can lie in a defect control valve, or in the cylinder load that is too high, or in other things. Diagnosing a hydraulic circuit means to identify the component or, as the case may be, the set of components that are defect and that are responsible for the observed misbehavior.

Usually, observed misbehavior is described qualitatively, since the impact of a defect component like a congested valve or a leaking pipe can only be defined in a simplified fashion. A simulation model of a hydraulic plant should reflect that fact, hence using only a handful of flow, pressure, and velocity values (`very_high`, `high`, `low`, `slow`, or ++, +, o, −, −−, etc.).

A qualitative algebra, which typically is defined on a small universe like above, is not powerful enough to form the basis for a complex behavior simulation, if pressure drop behavior[3] and circuits with feedback structures are to be simulated. Note that with respect to a qualitative simulation of a *single* component a sufficiently powerful algebra of qualitative derivatives and proportionalities can be stated. However, when connecting components to even small circuits, the ambiguity during qualitative simulation will result in a search space that cannot not be treated efficiently.

To overcome, or, at least, to noticeably alleviate the typical problems of a qualitative simulation of a continuous system, following concepts are pursued here:

1. For a circuit S an exact numerical simulation is performed. The simulation is based on realistic differential-algebraic behavior descriptions and provides information respecting flow directions and order of magnitudes for all physical quantities.

2. Those physical quantities that are necessary for a circuit diagnosis are used to define qualitative algebras for pressure, flow, and velocity.

3. *Individual* qualitative behavior laws are set up for each component of the interesting circuit. These behavior laws usually will not suffice the no-function-in-structure-principle, but work only for qualitatively simulating S.

4. The diagnosis process is organized hierarchically. Each series connection of valves and cylinders is replaced by a single substitute resistance followed by a single load-element. If during the diagnosis process one of the substitute components moves into the focus, the series connection is expanded and simulated on its own.

[3] The pressure drop at a valve, Δp, is proportional to the square of the flow, Q, through the valve.

1.2 Formulating a Diagnosis Hypothesis as an Instance of SAT

Based on the qualitative behavior laws that have been "compiled" from the numerical simulation of a particular circuit S, a propositional formula ψ can be set up. This formula encodes, among others, a set of qualitative behavior descriptions for S in the following way: If $\mathcal{I}$ is an interpretation that fulfills ψ, then $\mathcal{I}$ defines uniquely the physically correct behavior; i. e., it defines a reasonable quantity specification for S.

In the following the process of compiling the behavior for a component is demonstrated at a small example. Assume that we are given a circuit as shown in Figure 3 and that a propositional formula for the behavior of the marked pipe (thick line) is to be derived.

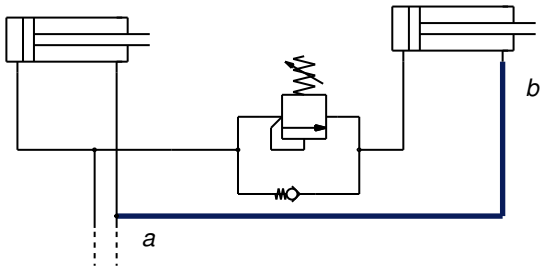

Figure 3. Hydraulic subcircuit with two coupled cylinders.

1. A qualitative formulation of the pipe behavior in first order logic is defined as follows:

   $$pipe(x) \rightarrow [ok(x) \rightarrow p_a(x) = p_b(x) \ \wedge \ Q_a(x) = Q_b(x)],$$

   where the equations define an equal potential and a mass balance constraint respectively.

2. Furthermore, let us assume that pressure values between 0 and 10 Bar and flow values between 0 and $24\ l/min$ have been computed within the course of a numerical simulation. Since this pipe leads to a sink, the simulated pressure values are rather low and can be mapped onto the two qualitative values `zero` and `low`. The flow values are mapped onto the qualitative values `zero`, `very_low`, and `low`.

   Note that a qualitative pressure value of `low` may be mapped to a higher numerical pressure value, when describing the leftmost pipe in the above circuit.

3. Within a qualitative algebra, $\oplus_{pipe}$, it is defined in which way the confluence of two pipes has to be computed. For instance, `very_low` $\oplus_{pipe}$ `low` = `low`.

4. From the description in first order logic along with the qualitative algebra a propositional behavior formula is instantiated. Below, such formula is shown as a piece of generated code in LISP syntax. The pipe is connected to the nodes `a` and `b` in the circuit, the qualitative values `zero`, `very_low`, and `low` have been abbreviated with 0, 1, and 2 respectively. Note that `Pa=1`, for instance, does not describe an assignment operation but is merely a variable name.

```
(:OR (:NOT PIPE-a-b_IS_OK)
  (:AND Pa=1 Pb=1 Qa=2 Qb=2) (:AND Pa=1 Pb=1 Qa=1 Qb=1)
  (:AND Pa=1 Pb=1 Qa=0 Qb=0) (:AND Pa=0 Pb=0 Qa=2 Qb=2)
  (:AND Pa=0 Pb=0 Qa=1 Qb=1) (:AND Pa=0 Pb=0 Qa=0 Qb=0))
```

The complete diagnosis hypothesis ψ of a circuit S consists of the logical conjunction of the following elements:

- behavior descriptions for all components in S
- unification constraints according to the topology of S
- cardinality constraints for physical quantities
- assumptions for the component's failure modes
- observations made at the real system

Depending on the pressure and flow resolution that has been chosen, a diagnosis hypothesis ψ for the circuit in Figure 3 contains between 1500 and 10000 variables.

2 Learning an Analysis Order for SAT

2.1 On Solving SAT

The solution of combinatorial problems by encoding them as propositional formulas and testing these formulas for satisfiability is a well-known approach in complexity theory. Examples of such encodings are given in nearly all textbooks; a more extensive collection can be found in [16]. Also, the problem library TPTP contains application problems formulated as logical formulas that can be used for testing and evaluating automated theorem provers [19]. The authors in [7] describe how satisfiability is successfully used to improve the detection of faults in combinational circuits. All these approaches consider single formulas that are processed one at a time. Thus, the complexity of the formula is similar to the complexity of the problem itself.

In [12], the use of propositional satisfiability testing is shown for model generation for first order formulas. Their approach also leads to sets of propositional formulas that need to be tested. However, the authors evaluate the formulas one after the other, guided by a fixed model generation process.

In our application we are confronted with large sets Ψ of propositional formulas each of which defining a particular diagnosis hypothesis. The formulas $\psi \in \Psi$ describe the same hydraulic system and differ only in partial truth assignments, which encode fault assumptions and observations. Solving the satisfiability problem for each formula is

rather simple, but the task is to find a subset of satisfiable formulas in these sets.

We apply the well-known linear transformations to generate corresponding formulas in CNF (equivalent with respect to satisfiability) and to analyze the CNF according to the given partial truth assignment. Analyzing the resulting formulas ψ for satisfiability can be done most effectively by Davis-Putnam-algorithms. Since the structure of our formulas has similarities with the quasi group problems described in [12], the satisfiability tester SATO as well as its descendants like BerkMin are an efficient decision procedure as well [20, 9]. Interestingly, due to the simple structure of the formulas (despite of their size), some famous algorithms perform bad, though they work well on benchmark tests originating from the constant clause length model.

2.2 The Regression Problem

In order to speed up the entire analysis process, we need a method that sorts the formulas according to their likeliness of being satisfiable. This would allow us to investigate the most promising formulas first.

Formally, a function $p : \Psi \to R$ is required, where Ψ denotes the set of formulas, and $p(\psi)$ states for a formula $\psi \in \Psi$ the likeliness that ψ is satisfiable. This subsection outlines how p can be learned from a set of already analyzed formulas.

For each formula $\psi \in \Psi$ a set of features $\mathbf{d}(\psi) = (d_1, \ldots, d_p)$ is generated. Typical features for a formula are the number of literals, the ratio between the number of clauses and variables, or graph-based features; they are listed in the next subsection. Features should indicate whether a formula is satisfiable, while at the same time the feature computation must be significantly easier than the satisfiability analysis.

The function p can be approximated by a function $\hat{p} : R^p \to R$, mapping for each formula ψ from the feature vector $\mathbf{d}(\psi)$ onto the likeliness of ψ being satisfiable.

Given a set of typical formulas, Ψ_{learn}, whose satisfiability is known, $\hat{p}$ can be learned by standard regression techniques. Applying regression and learning $\hat{p}$ forms a preprocessing step, therefore runtime considerations are less important here. For each formula $\psi' \in \Psi_{learn}$ the feature vector $\mathbf{d}(\psi')$ and $\hat{p}(\psi')$ is calculated. $\hat{p}(\psi')$ is computed as follows:

$$\hat{p}(\psi') = \begin{cases} 1 & \text{if } \psi' \text{ is satisfiable,} \\ 0 & \text{else} \end{cases}$$

For runtime reasons a neural network is used to approximate the solution to the regression problem. The reader may refer to [2, 15, 13, 10] for further details about neural networks and regression respectively. Since for each formula $\psi' \in \Psi_{learn}$ the input to the network, $\mathbf{d}(\psi')$, and the correct output, $\hat{p}(\psi')$, is known, a supervised learning strategy can be applied. In our experiments a neural network with two hidden layers and, as learning function, standard back-propagation were used.

Note that though for all formulas $\psi' \in \Psi_{learn}$ the function $\hat{p}(\psi')$ has only the values 0 or 1, $\hat{p}(\psi)$ can result in values not equal to 0 or 1 for a formula $\psi \notin \Psi_{learn}$.

2.3 Feature Generation

Crucial for the success of the learning process is the generation of features $\mathbf{d}(\psi) = (d_1, \ldots, d_p)$ for a formula ψ. The features used here fall into three different categories:

- *Statistical Features.* These features comprise formula properties like number of variables, formula length, number of literals, number of positive / negative literals, number of clauses, or average negative literal occurrence per variable. For formulas with a constant clause length it is known, that the ratio of number of clauses and number of variables is important. This feature is also used here, even though the clause length is not constant.

- *Logic Features.* By applying resolution to the formula, a deeper insight into its structure can be gained. In particular, the number of non-tautological resolvents proved to be valuable.

- *Graph Features.* In order to analyze the structure of a formula, the following graph is constructed: Each literal and each clause is represented as a node. Literal nodes are connected to a clause node if the literal is a member of that clause. All clauses are connected by an additional top node. Figure 4 shows the graph of the formula $\alpha = (A \vee B \vee \neg C) \wedge (\neg C \vee D \vee \neg E)$.

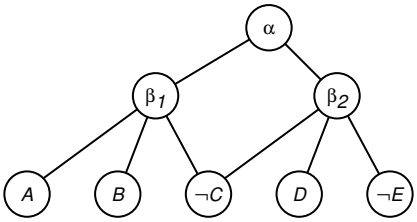

Figure 4. A graph of a formula α that is in CNF.

The following graph properties are used as formula features:

  - maximum, minimum, and average distance between nodes
  - maximum, minimum, and average node degree
  - λ-value: This value measures the connectivity within clusters in the graph; the clusters are defined implicitly by the λ-value.

In order to find the most informative features, an evaluation of both the importance of a single feature and the dependency between features is needed. Several algorithms

exist to rate feature importance, two are used here. Details about feature selection can be found in [1, 3, 11].

By calculating the correlation between two features, dependencies can be found. The correlation between a feature and the satisfiability is a hint for the feature's importance. Drawbacks of this method are, that only dependencies between two features are discovered and that not all dependencies are identified.

A neural network also rates feature importance. In a 2- or 3-layer network even feature *interactions* are taken into consideration. A key problem is the extraction of this knowledge. If the neural network consists of a single perceptron only, the feature rating problem becomes much easier.

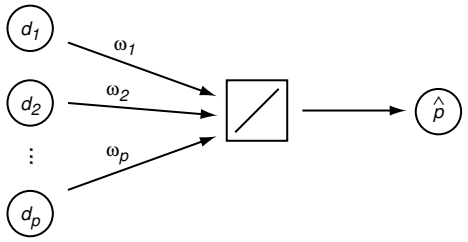

Figure 5. A single perceptron.

Figure 5 shows such a network. Each feature d_i is directly connected to the perceptron. A learning algorithm, e. g. backpropagation, optimizes the weights ω_i. ω_i is a hint for a feature's importance: features with large values support the satisfiability of the formula, features with negative weight contradict a possible satisfiability. While features with weights close to 0 are rather unimportant, a feature's importance can not be concluded from a large weight. Moreover, features that are only important in combination with other features cannot be discovered this way.

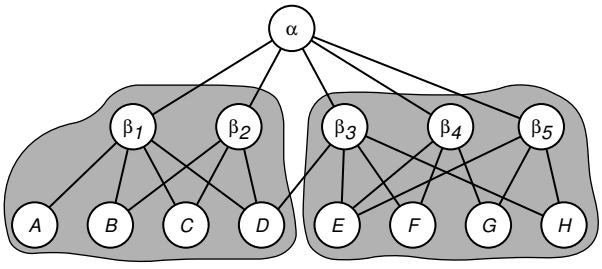

Figure 6. A clustered formula graph.

Nevertheless, by combining those two methods, hints for the importance of features can be found. By leaving out probable unimportant features, learning $\hat{p}$ again and comparing the error rate with and without those features, a set of important features can be identified.

Within our experiments, the exclusive use of the statistical features proved to be insufficient. Including the logic features as well as the graph features helped to predict the satisfiability. This can be illustrated exemplary with the following formula $\alpha = (A \vee B \vee C \vee D) \wedge (B \vee C \vee D) \wedge (D \vee E \vee F \vee H) \wedge (E \vee F \vee G) \wedge (E \vee G \vee H)$. Figure 6 shows the corresponding graph. The node degree depends mainly on the size of the formula and on the frequency of literal usage. The average distance in a graph is also a measure for the degree of connectivity between literals.

Observe that with growing connectivity of the graph the probability of satisfiability decreases. This makes sense from a logic point of view, since contradicting clauses are more probable when clauses have many literals in common.

3 Results

Several small and medium-sized circuits have been diagnosed using our approach, a small example can be seen in Figure 7.

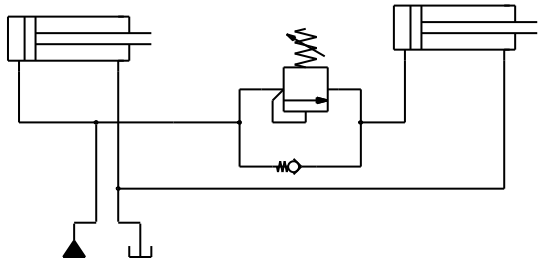

Figure 7. Hydraulic circuit with two coupled cylinders.

An algorithmic description of the diagnosis process is given in the following box.

Input. Formulas $\Psi = \{\psi_1, \ldots, \psi_n\}$ of diagnosis hypotheses.
Output. A diagnosis $\psi \in \Psi$

function diagnose (Ψ)
(1) **if** only one formula $\psi \in \Psi$ is satisfiable,
then return ψ as the diagnosis,
(2) **else** ask user whether o can be observed.
(3) **if** o is true
then $\Psi' = \{\psi \wedge o \mid \psi \in \Psi\}$
else $\Psi' = \{\psi \wedge \neg o \mid \psi \in \Psi\}$
(4) diagnose(Ψ')

Under the single-fault-assumption, the size of the set of hypotheses, $|\Psi|$, corresponds to the number of components in S.

The classification function $\hat{p}$ is used as follows. In Step (2) the objective is to choose an observation o that discriminates most between the hypotheses $\psi \in \Psi$, say, that leaves a minimum number of formulas satisfiable. Let $O = \{o_1, \ldots, o_m\}$ denote the set of all possible observations. Then the best observation $o \in O$ can be characterized as the maximum of the function $\theta(o)$:

$$\theta(o) = p(o) \cdot \big|\{\psi \wedge o \text{ is contradictory} \mid \psi \in \Psi\}\big|,$$

where $p(o)$ defines the probability for the occurrence of o.

To optimize the selection of the next measurement, $\mathcal{O}(|\Psi| \cdot |O|)$ formulas have to be analyzed. By using $\hat{p}$ as

an estimator for the satisfiability test, this runtime behavior can be significantly improved.

Note that the heuristic function $\hat{p}$, when applied to guide the search within Step (2), does not affect the correctness of a found diagnosis. However, when using $\hat{p}$ as a heuristic for the satisfiability test in Step (1), it cannot be guaranteed that the correct diagnosis is found.

To learn $\hat{p}$, observations were generated randomly and the resulting formulas were tested with respect to their satisfiability. For the circuit shown in Figure 7, 500 formulas formed the basis for learning. Afterward, $\hat{p}$ was able to correctly classify 75% of the newly presented test formulas.

Our current research is threefold and focuses onto the following points:

- refinement of the diagnosis model by differentiating between a larger number of components
- identification and evaluation of new features by which a diagnosis hypothesis ψ can be better characterized
- verification of the approach in other domains

References

[1] H. Almuallim and T. G. Dietterich. Efficient Algorithms for Identifying Relevant Features. In *Proceedings of the 9th Canadian Conference on Artificial Intelligence*, pages 38–45, Vancouver, 1992. Morgan Kaufmann.

[2] R. Beale and T. Jackson. *Neural Computing, an introduction*. Institute of Physical Publishing, Bristol and Philadelphia, 1994.

[3] M. Dash and H. Liu. Feature Selection for Classification. *Intelligent Data Analysis*, 1 (3), 1997.

[4] J. de Kleer, A. Mackworth, and R. Reiter. Characterizing Diagnoses and Systems. *Artificial Intelligence*, 56:197–222, 1992.

[5] J. de Kleer and B. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32:97–130, 1987.

[6] J. de Kleer and B. Williams. Diagnoses with Behavioral Models. In *Proc. of 11th IJCAI*, pages 1324–1330, 1989.

[7] F. Ferguson and T. Larrabee. Test pattern generation for realistic bridge faults in CMOS ICs. In *International Test Conference*, pages 492–499, Altoona, Pa., USA, Oct. 1991. IEEE Computer Society Press.

[8] K. Forbus and J. de Kleer. *Building Problem Solvers*. MIT Press, Cambridge, MA, 1993.

[9] E. Goldberg and Y. Novikov. Berkmin: A fast and robust sat solver, 2002.

[10] D. W. Hosmer and S. Lemeshow. *Applied Logistic Regression*. John Wiley & Sons, New York, 1989.

[11] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant Features and the Subset Selection Problem. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, San Francisco, CA, 1994. Morgan Kaufmann.

[12] S. Kim and H. Zhang. ModGen: Theorem proving by model generation. In *Proceedings of the 12th National Conference on Artificial Intelligence. Volume 1*, pages 162–167, Menlo Park, CA, USA, July 31–Aug. 4 1994. AAAI Press.

[13] R. H. Myers. *Classical and Modern Regression with Applications*. Duxbury Press, Boston, 1986.

[14] R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, Apr. 1987.

[15] W. S. Sarle. Neural Networks and Statistical Models. In *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, pages 1538–1550, Cary, NC, USA, 1994. SAS Institute Inc.

[16] H. Stamm-Wilbrandt. Programming in propositional logic or reductions: Back to the roots (satisfiability). Technical Report IAI-TR-93-3, Department of Computer Science, University of Bonn, Mar. 1 1993. Tue, 10 Oct 98 00:00:00 GMT.

[17] M. Stefik. *Introduction to Knowledge Systems*. Morgan Kaufmann Publishers, Inc., 1995. pp. 670–771.

[18] P. Struss and O. Dressler. "Physical Negation" - Integrating Fault Models into the General Diagnostic Engine. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan, USA*, volume 2, pages 1304–1321, 1989.

[19] C. B. Suttner and G. Sutcliffe. The TPTP problem library — v2.1.0. Technical Report JCU-CS-97/8, Department of Computer Science, James Cook University, 15 Dec. 1997.

[20] H. Zhang. SATO: An efficient propositional prover. In W. McCune, editor, *Proceedings of the 14th International Conference on Automated deduction*, volume 1249 of *LNAI*, pages 272–275, Berlin, July 13–17 1997. Springer.