Critical Reasoning

Olivier Raiman, Johan de Kleer and Vijay Saraswat
Xerox Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304 USA
Email: {dekleerjaiman^araswatliSparc.xerox.com

Abstract

Model-based diagnosis algorithms face a combinatorial explosion. To combat this explosion, this paper presents a fundamentally new architecture, IMPLODE, which constructs an abstract representation of the environment, the conflict, and the diagnosis spaces using a sensitivity analysis of assumptions. Experimental results show that the most dramatic improvement is obtained for circuits which are the most difficult to diagnose using previous algorithms. Moreover, typical sources of combinatorial explosion, such as reconvergent fanout, are a source of combinatorial implosion for IMPLODE.

1 Combinatorial Explosion

Model-based diagnosis engines [de Kleer and Williams 87; de Kleer 91; Reiter 87] face a potential combinatorial explosion of:

1. the environment space,
2. the conflict space,
3. the diagnosis space.

This paper presents a new architecture, IMPLODE, which requires neither a candidate generator nor checking candidates for consistency. IMPLODE uses a sensitivity analysis of assumptions to assign a criticality level to assumptions. IMPLODE keeps track of the assumptions* criticality level, and builds up an abstract representation of the environment, conflict and diagnosis spaces, using a database of:

1. critical environments,
2. critical conflicts,
3. critical diagnoses.

IMPLODE avoids the combinatorial explosion of previous generations of model-based diagnostic engines. Experimental results show that the most dramatic improvement is obtained for the circuits which are the most difficult to diagnose using previous diagnosis engines. Moreover, typical sources of combinatorial explosion, such as reconvergent fanout, are a source of combinatorial implosion for IMPLODE.

2 Sensitivity Analysis

One of the earliest model-based diagnosis engines is SOPHIE [de Kleer and Brown 92]. SOPHIE utilizes a form of sensitivity analysis based on partitioning the set of assumptions supporting a prediction into two sets:

1. The set of primary assumptions, P,
2. The set of secondary assumptions, S.

SOPHIE's criterion for determining whether an assumption is primary or secondary is based on a form of order of magnitude reasoning. An assumption which significantly contributes to the magnitude of the predicted value is primary otherwise it is secondary. In other words, the magnitude of the predicted value is extremely sensitive to its primary assumptions. One of SOPHIE's rules which exploits this distinction restricts the hypothesis space as follows: *// two environments* $\langle P_1, S_1 \rangle$, *and* $\langle P_2, S_2 \rangle$, *support the same prediction, then all the components which belong to* $[P_1 \cup P_2] - [P_1 \cap P_2]$ *are exonerated.* The intuition behind the rule is as follows. If an assumption which significantly contributes to the predicted value is violated, then the predicted value would be significantly different. Thus, assumptions not in common between the antecedent primary assumption sets are exonerated.

SOPHIE's use of sensitivity analysis for model-based diagnosis is based on a number of presuppositions which limit its generality:

1. Environment space: The criterion used to distinguish primary and secondary assumptions is specific to the domain of analog circuits.
2. Conflict space: The partitioning into primary and secondary assumptions is not extended to conflicts — SOPHIE does not distinguish between primary and secondary assumptions in conflicts.
3. Diagnosis space: The exoneration rule is based on the single-fault assumption.

3 Critical Environments and Abstractions

The assumption set *A* contains $ok(C_J)$ for every component C_J. The description of the system to diagnose and the observation set are given by (effectively) a propositional theory *T* over a set of literals £. We presume that 7 is consistent. A node n designates a literal, that is an atom or its negation.

An environment *E* supporting a literal n is a set of assumptions such that:

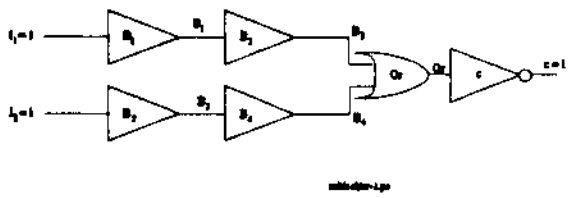

Figure 1: $\{ok(Or)\}$ is the primary assumption for $Or = 1$.

1. [Entailment]: $T \cup E \models n$.
2. [Consistency]: $T \cup E$ is consistent.

The set of environments supporting a literal is noted $\mathcal{E}(n)$. $\mathcal{E}(n)$ is characterized by its minimal (with respect to set inclusion) elements, called minimal environments. However these minimal environments are computationally prohibitive to construct. As an alternative, the following definition of critical environment introduces an abstraction which can be used to characterize $\mathcal{E}(n)$ far more parsimoniously.

Definition 3.1 (Critical Environment) The critical environment for node n relative to $S \subseteq A$ is:

$$\hat{S}(n) = \bigcap \{E \in \mathcal{E}(n) \mid E \subseteq S\} \tag{1}$$

□

Here and in the following we shall assume that there is a special assumption, called `false` which takes on the value false in all models, and that $\bigcap \emptyset \stackrel{def}{=} \{\texttt{false}\}$.

Example 3.1 Consider the circuit, (see Figure 3), consisting of an or-gate, Or, two rows of buffers, $\{B_1, B_2\}$, $\{B_3, B_4\}$, and an inverter, C, at the output of the or-gate. Let S be the entire set $\mathcal{A}$.

Given, the observations $\{i_1 = 1, i_2 = 1\}$, there are two minimal environments supporting the prediction $Or = 1$:

$$\{ok(B_1), ok(B_2), ok(Or)\}, \quad \{ok(B_3), ok(B_4), ok(Or)\}. \tag{2}$$

$ok(Or)$ is the only assumption which is common to these two environments. Therefore the critical environment for $Or = 1$ relative to $\mathcal{A}$ is $\{ok(Or)\}$. □

Notice that the critical environment for a literal n relative to S is always included in the minimal environments for n in S. If a node has a single minimal environment E in S, then E is also the critical environment for n relative to S.

The abstraction process leading to the generation of critical environments ensures the following local consistency property for the critical environment any given node:

Proposition 1 *$T \cup \hat{S}(n)$ is consistent if T is consistent and $\hat{S}(n)$ is consistent.*

Proof 1 $\hat{S}(n)$ is an intersection of environments which by definition already possess this consistency property (if it is consistent, i.e., does not contain `false`). □

Thus critical environments inherit the consistency property of environments. However, in general it is not true that critical environments inherit the entailment property of environments, i.e. in general it is not true that $T \cup \hat{S}(n) \models n$. In general, the critical environments only possess the entailment property if the background theory T is augmented adequately. But augmenting T may lead to inconsistencies. The following sections introduce the concept of critical conflicts and critical diagnoses to determine how to augment T so that the critical environments become environments of the augmented theory, and show how to exploit the augmented theory for diagnosis purposes.

4 Critical Conflicts

A conflict C is a set of assumptions such that $C \cup T$ is inconsistent. A minimal conflict has no other as a subset. As the number of conflicts also explodes we exploit similar intuitions to those underlying critical environments to introduce the notion of critical conflicts.

Definition 4.1 (Critical Conflicts) The critical conflict for $S \subseteq \mathcal{A}$ is:

$$\hat{S}(\bot) = \bigcap \{C \text{ a conflict} \mid C \subseteq S\} \tag{3}$$

□

Example 4.1 (Continued) If $C = 1$ the minimal conflicts are:

$$\begin{array}{l} \{ok(B_1), ok(B_2), ok(Or), ok(C)\} \\ \{ok(B_3), ok(B_4), ok(Or), ok(C)\} \end{array} \tag{4}$$

Thus:

$$\hat{\mathcal{A}}(\bot) = \{ok(Or), ok(C)\} \tag{5}$$

□

Since the minimal conflicts characterize all conflicts (via subset), we can always use the minimal conflicts to stand for all the conflicts in our definitions and examples. This example shows that a critical conflict relative to S subsumes all the conflicts in S.

Critical conflicts offer an abstract representation of the conflict space. Critical conflicts are also the keystone for identifying whether or not a critical abstraction which abstracts simultaneously the environments for multiple nodes preserves the consistency of the background theory T. More precisely, a critical abstraction is defined as follows:

Definition 4.2 (Critical Abstraction) The critical abstraction of a theory T relative to $S \subseteq \mathcal{A}$ is the theory:

$$\hat{S}(T) = T \cup \{\hat{S}(n) \rightarrow n \mid n \in \mathcal{L} \cup \{\bot\}\} \tag{6}$$

□

Note that the process of abstraction adds only Horn clauses to the original theory; in particular the abstraction of a Horn theory is a Horn theory.

Proposition 2 (Consistency of Critical Theories) *$\hat{S}(T)$ is consistent iff $\hat{S}(\bot) \neq \emptyset$.*

Proof 2 Assume that $\hat{S}(\bot) = \emptyset$. Since $\hat{S}(\bot) \rightarrow \bot$ belongs to $\hat{S}(T)$, $\hat{S}(T)$ is inconsistent.

Assume that $\hat{S}(\bot) \neq \emptyset$; choose $A \in \hat{S}(\bot)$. Since A belongs to all the conflicts in S it follows that: $T' \stackrel{def}{=} T \cup (S \backslash \{A\}) \cup \{\overline{A}\}$ is consistent. To prove that $\hat{S}(T)$ is consistent, it suffices to prove that

$$T' \models \hat{S}(n) \rightarrow n \tag{7}$$

for each node n. This follows trivially if $A \in \hat{S}(n)$. If not, there is an environment $E \subseteq (S \setminus \{A\})$ for n. But then $T \models (S \setminus \{A\}) \rightarrow n$, so that $T' \models n$. □

In many cases $\hat{\mathcal{A}}(\bot)$ will be non-empty — all the conflicts will have a non-empty intersection. For instance, this will happen if the system obeys a single-fault assumption. In this case $\{\overline{A}\}$ is a minimal diagnosis for every $A \in \hat{\mathcal{A}}(\bot)$.

However, it could be that $\hat{\mathcal{A}}(\bot) = \emptyset$. In such a case there must be at least two mutually disjoint conflicts, C_1 and C_2 in $\mathcal{A}$. Thus, we can partition $\mathcal{A}$ into two subsets S_1 (containing C_1) and S_2 (containing C_2). We may then exploit the intuition that often designed systems are constructed in such a way that they are *nearly decomposable*. So we may choose to transform the initial search for diagnoses into two *separate* diagnosis problem: we consider the assumptions in S_1 separately from those in S_2, constructing separately the critical environments and conflicts for these two sets. Once that is done, we desire to glue together information about the diagnosis of the system gleaned from these two disjoint analyses.

In general, however, $\hat{S}_1(T) \cup \hat{S}_2(T)$ is not guaranteed consistent even if separately both $\hat{S}_1(T)$ and $\hat{S}_2(T)$ are consistent. (Examples are easy to construct.) We now give certain rather general conditions under which the consistency of conjunctions of critical abstractions is guaranteed.

Definition 4.3 (Critical Cover) A critical cover relative to a subset S of $\mathcal{A}$ is a set $\{S_1, \ldots, S_k\}$ of subsets of S such that:

Critical consistency: $\hat{S}_i(\bot)$ is non-empty, for all i.

Covering: Every conflict $C \subseteq S$ is subsumed by $\hat{S}_i(\bot)$, for some i.

Disjointness: $\hat{S}_i(\bot)$ is disjoint from S_j, for every distinct i and j.

□

In such a case, the set $\{\hat{S}_1(\bot), \ldots, \hat{S}_k(\bot)\}$ is called a *covering set of critical conflicts*. Note that the disjointness condition implies that $\hat{S}_i(\bot)$ is distinct from $\hat{S}_j(\bot)$ — hence any cover contains at most $|S|$ critical conflicts.

Proposition 3 (Consistency of critical covers.) *If S is a critical cover relative to $\mathcal{A}$ then*

$$\bigcup\{\hat{S}_i(T) \mid S_i \in \mathcal{S}\} \tag{8}$$

is consistent.

Proof 3 The proof is very similar to that of Theorem 2. Since each $\hat{S}_i(\bot)$ is non-empty, choose an $A_i \in \hat{S}_i(\bot)$. Consider the augmented theory

$$T' \stackrel{def}{=} T \cup \{\overline{A_1}, \ldots, \overline{A_k}\} \cup ((\cup_{i=1\ldots k} S_i) \setminus \{A_1, \ldots, A_k\}) \tag{9}$$

By Disjointness, T' is equivalent to

$$T \cup \{\overline{A_1}, \ldots, \overline{A_k}\} \cup (\cup_{i=1\ldots k}(S_i \setminus \{A_i\})) \tag{10}$$

First, we claim that T' is consistent. Suppose it is not. Then it must be the case that

$$T \models (\cup_{i=1\ldots k}(S_i \setminus \{A_i\})) \rightarrow (A_1 \vee \ldots \vee A_k) \tag{11}$$

However, if the S_i are not trivial (that is, contain at least one conflict), then for each i, there must be a conflict of the form $A_i, \delta_i \rightarrow$ where $\delta_i \subseteq S_i \setminus \{A_i\}$. Hyper-resolving the k such clauses against the clause in (11) yields the $\mathcal{A}$ conflict of T:

$$\cup_{i=1\ldots k}(S_i \setminus \{A_i\}) \rightarrow \tag{12}$$

However, this conflict is not subsumed by any $\hat{S}_i(\bot)$ (which must contain A_i), in violation of our Coverage assumption. The contradiction establishes that T' is consistent.

Now we show that for every literal n and i:

$$T' \models \hat{S}_i(n) \rightarrow n \tag{13}$$

This obviously follows if $A_i \in \hat{S}_i(n)$. Otherwise, there is some environment for n contained in $S_i \setminus \{A_i\}$; that is $T \models E \rightarrow n$, where $E \subseteq S_i \setminus \{A_i\}$. It follows then that $T' \models n$, and we are done. □

Critical covers do not always exist as the following example shows.

Example 4.2 The simplest example is given by the cyclic theory:

$$\begin{array}{l} a \wedge b \rightarrow \bot \\ a \wedge c \rightarrow \bot \\ b \wedge c \rightarrow \bot \end{array} \tag{14}$$

Take $S = \mathcal{A} = \{a, b, c\}$. The only subset of S for which there is a non-trivial intersection of conflicts is the set S — however, in this case the intersection is $\emptyset$. □

When critical covers do not exist, we are obliged to seek for maximal subsets of $\mathcal{A}$ which do have a cover. Exploring the space of weaker definitions of critical cover (that will guarantee the existence of critical covers in more cases) is beyond the scope of the paper. We expect to see significant tradeoffs between the weakness and computability of the definitions; indeed we would not be surprised if it is impossible to simultaneously achieve a definition of critical cover that guarantees that only minimal diagnoses are generated (see Theorem 6 below), that always exists for consistent theories, and that can be used to compute critical diagnoses efficiently and incrementally.

5 Critical Diagnoses

This section exploits the concept of criticality to obtain a concise representation of the diagnosis space. For a given set of literals S, the set $\{\overline{l} \mid l \in S\}$ is noted $\overline{S}$. We shall treat $\overline{\overline{S}}$ as identical to S. A diagnosis is a set $\overline{S}$ for some $S \subseteq \mathcal{A}$ such that $T \cup \overline{S}$ is consistent.

Definition 5.1 (Critical diagnosis) A critical diagnosis of T relative to a $S \subseteq \mathcal{A}$ is a set $\Delta \cap \overline{S}$ such that:

1. Δ is a diagnosis of $\hat{S}(T)$.
2. There is no other diagnosis Δ' of $\hat{S}(T)$ such that $\Delta' \cap \overline{S}$ is a strict subset of $\Delta \cap \overline{S}$.

□

In the following, by a critical diagnosis of T, we shall mean a critical diagnosis of T relative to $\mathcal{A}$.

Example 5.1 (Continued) Since $\hat{\mathcal{A}}(\bot) = \{ok(Or), ok(C)\}$, the critical diagnoses relative to $\mathcal{A}$ are the two potential single faults: $\{\overline{ok(Or)}\}$ and $\{\overline{ok(C)}\}$. □

Say that a diagnosis Δ *hits* a conflict C if there exists an $\overline{A} \in \Delta$ such that $A \in C$.

Proposition 4 *If $\hat{S}(\mathcal{T})$ is consistent, and there is at least one conflict $C \subseteq S$ of $\mathcal{T}$, then the set of critical diagnosis of $\mathcal{T}$ relative to S is*

$$\{\{\overline{A}\} \mid A \in \hat{S}(\bot)\} \tag{15}$$

Proof 4 Assume S contains at least one conflict of $\mathcal{T}$. Any diagnosis of $\hat{S}(\mathcal{T})$ must hit $\hat{S}(\bot)$; indeed any element of $\hat{S}(\bot)$ is adequate. A critical diagnosis can contain at most one such element. □

Note that if S does not contain any conflict of $\mathcal{T}$, then $\hat{S}(\bot) = \{\texttt{false}\}$, yielding the vacuous critical diagnosis $\emptyset$.

Example 5.2 (Continued) Since $\hat{\mathcal{A}}(\bot) = \{ok(Or), ok(C)\}$ Therefore the critical diagnoses relative to $\mathcal{A}$ are the two potential single faults: $\{\overline{ok(Or)}\}$ and $\{\overline{ok(C)}\}$. □

Proposition 5 *Every critical diagnosis of $\mathcal{T}$ relative to any $S \subseteq \mathcal{A}$ is a subset of a minimal diagnosis of $\mathcal{T}$.*

Proof 5 No proper subset of a critical diagnosis relative to S can hit all the conflicts of S. □

Theorem 6 (Diagnostic Hypercube) *Let $\{S_1, \ldots, S_k\}$ be a critical cover of $\mathcal{A}$ for $\mathcal{T}$. If Δ_i is a critical diagnosis for S_i, then $\Delta = \Delta_1 \cup \ldots \cup \Delta_k$ is a minimal diagnosis for $\mathcal{T}$.*

Proof 6 Since Δ hits every conflict in $\mathcal{A}$, it is a diagnosis. Suppose it is not minimal. Then there is an $\overline{A} \in \Delta$ such that $\Delta' = \Delta \setminus \{\overline{A}\}$ is a diagnosis. Note that there is exactly one $i \leq k$ such that $A \in \hat{S}_i(\bot)$. (To avoid trivialities, assume that A is distinct from $\texttt{false}$.) However, Δ' cannot hit any of the conflicts contained in S_i, since Δ' is disjoint from S_i. Hence Δ' cannot be a diagnosis. □

Thus finding a critical decomposition of $\mathcal{A}$, if any, allows a linear encoding of an exponential number of minimal diagnoses.

Example 5.3 (Continued) Since $\mathcal{A}$ forms a critical partition, $\mathcal{T}$ contains two single faults as minimal diagnoses: $\{\overline{ok(Or)}\}$ and $\{\overline{ok(C)}\}$. □

Proposition 7 *If $\mathcal{A}$ forms a critical partition then $\hat{\mathcal{A}}(\bot)$ contains all the single faults.*

Searching for critical conflicts enables an architecture for model-based diagnosis which ensures the consistency of the candidates while bypassing the interpretation construction and context switching of conventional diagnostic algorithms.

5.1 Critical Probes

For the purpose of this paper we assume that probing is the only kind of action available to differentiate among different diagnoses. We ignore component failure rates and cost of probes. The conventional (GDE-like) approach to finding the probe which differentiates best among all the diagnoses is to determine the environments for the different nodes.

Our strategy here is to focus on critical diagnoses and find a probe which differentiates best among the critical diagnoses. To identify this probe we would like to consider the nodes' critical environments. To justify ignoring the supersets of critical environments we show that once the diagnosis space is restricted to the set of critical diagnoses the critical environments possess the following entailment property:

Proposition 8 (Critical Entailment) *Let $S_1, \ldots, S_k$ be a critical partition of $\mathcal{A}$, and $\mathcal{P}$ be the product space of critical diagnoses, and n be a literal of $\mathcal{L}$.*

$$\mathcal{T} \cup \{\bigvee \Delta, \Delta \in \mathcal{P}\} \models \hat{S}_i(n) \rightarrow n \tag{16}$$

Example 5.4 (Continued) Since $\hat{\mathcal{A}}(Or = 1) = \{ok(Or)\}$ and $\hat{\mathcal{A}}(Or = 0) = \{ok(C)\}$, probing the output of the Or performs a half split on the space of critical diagnoses. Thus, if initially all the components are equally likely to fail and the cost of different probes are the same, this output of Or is the probe which differentiates best among the critical diagnoses. Notice also that, in this case, the output of Or is also the best probe to differentiate among all consistent diagnoses. □

As new observations are gathered, the evolution of critical environments, critical conflicts and critical diagnoses may be non monotonic if the new observation eliminates all the critical diagnoses.

Example 5.5 (Continued) The two critical diagnoses predict that B_2 should be 1. If the output of B_2 is observed to be 0, then all the critical diagnoses are eliminated and there is a non monotonic evolution of the critical diagnoses. If $B_2 = 1$, then there are two disjoint minimal conflicts: $\{ok(B_1), ok(B_2)\}$ and $\{ok(B_3), ok(B_4), ok(Or), ok(C)\}$. These two disjoint minimal conflicts become the covering set of critical conflicts. The new set of critical diagnoses is given by the product space:

$$\overline{\{ok(B_1), ok(B_2)\}} \times \overline{\{ok(B_3), ok(B_4), ok(Or), ok(C)\}} \tag{17}$$

□

This non monotonicity highlights the fact that critical reasoning is a particular form of non monotonic reasoning. Exploring the link between critical reasoning and other forms of non monotonic reasoning is beyond the scope of this paper.

6 IMPLODE Algorithm

Given a Horn propositional theory which has a covering set of critical conflicts, IMPLODE identifies the critical conflicts and computes critical environments, and concisely encodes the set of critical diagnoses.

IMPLODE is an incremental algorithm. Counterintuitively, it does not operate by maintaining a single global data structure of the current best approximation of a critical cover which is then incrementally updated. Much in the spirit of an ATMS, each environment also carries with it, locally, the smallest set of assumptions (discovered so far in the process) which makes the environment a critical one.

We expand the notation of Section 2 to include the node or assertion which it supports. The triple $\langle M, P, S\rangle$ denotes a clause whose literals are $M \cup \overline{P} \cup \overline{S}$. P is the set of primary assumptions, S is the set of secondary assumptions, and M is the remaining set of literals. $\langle \emptyset, P, S\rangle$ represents a conflict.

Each $\langle M, P, S\rangle$ represents a critical environment (or conflict). P is the critical environment of M relative to $P \cup S$. P and S are always kept disjoint: S is the set of assumptions that are needed to be combined with P which makes P a critical environment. More formally:

$$(S \hat{\cup} P)(M) = P \tag{18}$$

During execution IMPLODE is only representing approximations of critical environments and critical conflicts. At the conclusion of the algorithm (if it terminates successfully) the current set of critical conflicts form a critical cover.

IMPLODE takes as input the set of clauses T and produces new clauses using the following rules:

IMPLOSION:

$$\frac{\langle M, P_1, S_1\rangle \quad \langle M, P_2, S_2\rangle}{\langle M, P_1 \cap P_2, S_1 \cup S_2 \cup [P_1 \setminus P_2] \cup [P_2 \setminus P_1]\rangle} \tag{19}$$

RESOLUTION: If $m \in M$ unless $P_1 \cup P_2 = \emptyset$ and $M = \{m\}$.

$$\frac{\langle M, P_1, S_1\rangle \quad \langle \{\overline{m}\}, P_2, S_2\rangle}{\langle M \setminus \{m\}, P_1 \cup P_2, [S_1 \cup S_2] \setminus [P_1 \cup P_2]\rangle} \tag{20}$$

EXPLOSION:

$$\frac{\langle \{m\}, \emptyset, S_1\rangle \quad \langle \{\overline{m}\}, \emptyset, S_2\rangle}{\langle \emptyset, S_1 \cup S_2, \emptyset\rangle} \tag{21}$$

IMPLODE, incrementally applies these rules, to a restricted set of clauses, the *active* clauses. Active clauses are determined by the following rules. When IMPLODE receives a new clause, this clause is initially marked as active. A clause can become inactive by either being critically subsumed or by being suppressed. A clause $\langle M_1, P_1, S_1\rangle$ is critically subsumed if there is another active clause $\langle M_2, P_2, S_2\rangle$ such that $M_2 \cup P_2$ is a strict subset of $M_1 \cup P_1$ or just of $S\setminus$ alone. When IMPLODE finds multiple critical environments for a node, it systematically chooses only one of those critical environments to propagate via the resolution rule — the other critical environments are suppressed (inactivated). An inactive clause whose reason for inactivation itself is inactivated or removed is reactivated.

IMPLODE prioritizes rule applications by always running higher priority rules if they are applicable:

1. Reactivate clauses whose reason for inactivation has disappeared.
2. Inactivate any clauses which are now critically subsumed.
3. Apply EXPLOSION.
4. Apply IMPLOSION.
5. For any literal with more than one critical environment, suppress all but one by marking it inactive because of the chosen critical environment.
6. Apply RESOLUTION.

Note that in this algorithm the number of critical environments for a node can never explode. Even in examples (such as levels of alternating and/or gates) which cause exponential behavior for GDE-like algorithms for single faults IMPLODE performs well.

7 IMPLODE Example

T, the initial set of clauses contains the models for all the components and connections. For example, the buffer B_1 is modeled by the two clauses:

$$\begin{array}{l}\langle\{\overline{i_1 = 1}, B_1 = 1\}, \{ok(B_1)\}, \emptyset\rangle \\ \langle\{\overline{i_1 = 0}, B_1 = 0\}, \{ok(B_1)\}, \emptyset\rangle\end{array} \tag{22}$$

The connection i_1 is modeled by the single clause:

$$\langle\{\overline{i_1 = 0}, \overline{i_1 = 1}\}, \emptyset, \emptyset\rangle \tag{23}$$

The two observations are:

$$\langle\{i_1 = 1\}, \emptyset, \emptyset\rangle \quad \langle\{i_2 = 1\}, \emptyset, \emptyset\rangle \tag{24}$$

Repeated applications of the resolution rule produces the following clauses:

$$\begin{array}{l}\langle\{B_1 = 1\}, \{ok(B_1)\}, \emptyset\rangle \\ \langle\{B_2 = 1\}, \{ok(B_1), ok(B_2)\}, \emptyset\rangle \\ \langle\{Or = 1\}, \{ok(Or), ok(B_1), ok(B_2)\}, \emptyset\rangle\dagger \\ \langle\{B_3 = 1\}, \{ok(B_3)\}, \emptyset\rangle \\ \langle\{B_4 = 1\}, \{ok(B_3), ok(B_4)\}, \emptyset\rangle \\ \langle\{Or = 1\}, \{ok(Or), ok(B_3), ok(B_4)\}, \emptyset\rangle\dagger\end{array} \tag{25}$$

The Implosion rule produces:

$$\langle\{Or = 1\}, \{ok(Or)\}, \{ok(B_1), ok(B_2), ok(B_3), ok(B_4)\}\rangle* \tag{26}$$

This clause immediately inactivates the two clauses marked †. Resolution produces:

$$\langle\{C = 1\}, \{ok(Or), ok(C)\}, \{ok(B_1), ok(B_2), ok(B_3), ok(B_4)\}\rangle\ddagger \tag{27}$$

Now suppose we observed:

$$\langle\{C = 0\}, \emptyset, \emptyset\rangle \tag{28}$$

Resolution immediately produces the critical conflict:

$$\langle\emptyset, \{ok(Or), ok(C)\}, \{ok(B_1), ok(B_2), ok(B_3), ok(B_4)\}\rangle* \tag{29}$$

This critical conflict permanently inactivates the clause ‡. The resolution rule also produces:

$$\langle\{Or = 0\}, \{ok(C)\}, \emptyset\rangle \tag{30}$$

Now suppose we observe:

$$\langle\{B_2 = 0\}, \emptyset, \emptyset\rangle \tag{31}$$

Resolution produces the critical conflict:

$$\langle\emptyset, \{ok(B_1), ok(B_2)\}, \emptyset\rangle \tag{32}$$

This clause inactivates the clauses marked * and this now causes the clauses marked † to be reactivated except the first such clause is immediately inactivated again by the new critical conflict. The single activated clause resolves to produce the critical conflict:

$$\langle\emptyset, \{ok(Or), ok(B_3), ok(B_3), ok(C)\}, \emptyset\rangle \tag{33}$$

8 Experimental Results

We have adapted our [de Kleer 91] model-based diagnosis algorithm to implement IMPLODE. IMPLODE is sufficiently different, that an overall reimplementation would produce far more significant performance improvements. But we report preliminary results in the following table. This table provides the results on only a few instances of each circuit. All instances are examples which had been earlier determined to be the most difficult for the unoptimized focused algorithm [de Kleer 91] and therefore the timings are far worse than average for those circuits. Each run is of a single test vector and one symptomatic output. All of the circuits are from the test suite provided in [Brglez *et al.* 85]. The timings are obtained on a Symbolics XL 1200, and include the setup time for running the experiment (which is now the dominant cost). The table has two columns. The first contains the timing for the algorithm of [de Kleer 91] entitled "AAAI91", the second one contains IMPLODE's timing.

IMPLODE shows the greatest improvement on the circuits with significant redundancy and reconvergent fanout—something extremely difficult for previous algorithms. For example, circuit c6288 is a 16 by 16 bit parallel multiplier and manifests a great deal of reconvergent fanout.

Device	Gates	AAA91	IMPLODE
c432	160	2.3	.3
c499	202	7.9	.4
c880	383	6.2	.8
cl355	546	242	1.4
c1908	880	89~	8
c2670	1193	33,	3
c3540	1169	1545	6
c5315	2308	1215	7
c6288	3600	∞	8
c7552	3512	1028	14

References

[Brglez *et al.* 85] Brglez, F., and H. Fujiwara, A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN, distributed on a tape to participants of the Special Session on ATPC and Fault Simulation, Int. Symposium on Circuits and Systems, June 1985; partially characterized in F. Brglez, P. Pownall, and R. Hum, Accelerated ATPG and fault grading via testability analysis, *Proc. IEEE Int. Symposium on Circuits and Systems,* (June, 1985) 695-698.

[Dague *et al* 87] Dague, P., Deves, P. and Raiman, O., Troubleshooting: when modelling is the trouble, Proceeding of sixth national conference on Artificial Intelligence AAAI (1987).

[de Kleer and Williams 87] de Kleer, J., Williams, B.C., Diagnosing multiple faults, Artificial Intelligence 3297-130.

[de Kleer 91] de Kleer, J. Focusing on probable diagnoses, in Proceedings AAAI-91, Anaheim, CA (1991) 842-848.

[de Kleer 92] de Kleer, J., Optimizing Focusing Model-Based Diagnosis, in the third international workshop on diagnosis principles, October 92.

[de Kleer and Brown 92] de Kleer, J. and J.S. Brown, Model-based diagnosis in SOPHIE III, in: *Readings in Model-Based Diagnosis* edited by W. Hamscher, J. de Kleer, and L. Console, (Morgan Kauftnann, 1992).

[Reiter 87] Reiter, R., A theory of diagnosis from first principles, Artificial Intelligence 32 (1987) 57-95.