Diagnosing Dependent Failures – an Extension of Consistency-based Diagnosis

Jörg Weber and Franz Wotawa
Technische Universität Graz
Institute for Software Technology
8010 Graz, Inffeldgasse 16b/2, Austria
Tel: +43 316 873 5723, Fax: +43 316 873 5706
{jweber,wotawa}@ist.tugraz.at

Abstract
Most applications of model-based diagnosis rely on the assumption that components fail independently. However, dependent failures are quite common in some domains. In such domains, the failure of one component may cause the failure of other components. Hence, multiple faults are much more likely, and the minimal diagnoses may not contain all components which have failed. In this work we propose the concept of diagnosis environments (DEs). DEs distinguish between independent and dependent failures, and they may consider components as failed which are not part of the minimal diagnoses. We provide a formalization of our approach and present an algorithm which computes all minimal DEs.

Introduction
The applications of model-based diagnosis (MBD) usually rely on the assumption that components fail independently (Reiter 1987)(de Kleer & Williams 1987). However, as de Kleer remarks in (de Kleer 1990), dependent failures are quite common in some domains. In such domains the failure of one component may lead to the failure of other components as well. A component $c_i$ fails in dependence of another component $c_j$ if (1) $c_i$ fails and (2) the failure of $c_i$ causes some "damage" to $c_j$ which prevents $c_j$ from behaving as desired and (3) the damage to $c_j$ is permanent, i.e., $c_j$ exhibits abnormal behavior even after a repair of $c_i$, and so $c_j$ must be repaired as well. If these 3 conditions hold, then we consider $c_j$ as abnormal, even though its failure has been caused by external circumstances instead of an internal fault.

For example, from our experience with diagnosis in mobile autonomous robots we know that dependent failures often occur in hardware-software hybrid systems. There are different reasons for this. First, software components may be tightly coupled with hardware components, and so a failure of the latter may have a fatal impact on the former. A typical example is the failure of a camera which causes a crash of the image processing software. Another reason is that software components are often based on the same underlying framework (e.g., CORBA), and a failure in the framework may propagate to those components. A third reason is the communication over channels which do not strictly decouple the involved components. E.g., when a component attempts to make a remote procedure call to a failed component, then the former component may also fail.

A more detailed discussion of dependent failures in the hardware-software hybrid system of a mobile autonomous robot is provided in (Weber & Wotawa 2007). A component which has failed due to an internal error is an independently failed component, whereas a component whose failure has been caused by another component is a dependently failed component. A dependently failed component may cause the failure of further components; i.e., there can be a cascade of dependent failures.

Dependent failures may also occur in physical systems: suppose a component $c_i$ fails and produces extra heat (or current, pressure, etc.) which causes physical damage to an adjacent component $c_j$. Due to the assumption that components fail independently, most MBD systems compute only minimal diagnoses. However, in the next section we will give an example which shows that, in case of dependent failures, the minimal diagnoses often do not contain all components which have failed. Furthermore, the knowledge of the causal order of failures is often crucial for a successful repair of the system.

In order to tackle these problems, we propose the concept of diagnosis environments (DEs) and minimal diagnosis environments (MDEs). Each DE corresponds to a (in general not minimal) diagnosis; i.e., it may consider components as independently or dependently failed even if they are not abnormal in all minimal diagnoses. Moreover, a DE states the causal order in which the components have failed by distinguishing between independent and dependent failures.

MBD systems employ a system description and observations in order to compute the diagnoses. In addition, our approach relies on a failure dependency graph which indicates possible failure dependencies. This graph can be seen as part of the system model. As it contains all possible dependencies (i.e., it considers the worst case), we need to dynamically determine which of the possible dependent failures may actually have occurred. For this purpose, we introduce the dependent failure description which is a logical description of system behavior in case of dependent failures. This description is used to eliminate implausible DEs.

To the best of our knowledge, our proposal is the first work within the field of consistency-based diagnosis which addresses the issue of dependent failures at the level of log-
ical reasoning. In (Weber & Wotawa 2007) we introduced the idea of DEs and provided examples from the diagnosis in mobile autonomous robots. We also presented the results of case studies in which the independent failure of one component caused the failure of many other components. Our diagnosis system computed the MDEs and was able to successfully repair a large part of the system, whereas a repair based solely on minimal diagnoses would not have succeeded in this situation.

The main contribution of this paper is a formalization of our proposal which is application-independent and based upon Reiter’s diagnosis framework (Reiter 1987). We also prove some important consequences of our definitions. Furthermore, we provide an algorithm which computes all MDEs of a system.

The following section provides an informal introduction to our proposal by means of an example. Then we formalize our approach and provide an algorithm which computes all MDEs for a system. Finally we provide a discussion and relate our research to the work of others.

Motivation

Figure 1: Software components of a robot control system.

Figure 1 depicts several software components of the control system of a mobile autonomous robot. The directed connections indicate data flows between the components. Observable connections are depicted by solid lines, whereas dotted lines represent unobservable ones. The data on unobservable connections cannot be monitored by the diagnosis system.

BD is a service which controls the hardware bus and receives/transmits data from/over the bus. LS and SS rely on BD to receive data from sensors which are connected to the bus. These components process the raw sensor data and dispatch the results over their output connections, ls_{o} and ss_{o}. Finally, GS controls the grabber arm of the robot. The connections between BD and the three other components are not observable, because the latter perform remote calls to procedures of BD in order to receive/transmit data, and these calls cannot be intercepted.

As usual in MBD, we provide a logical system description (SD) which captures the nominal behavior of components. The predicate \(ab\) denotes abnormal:

\[
\neg ab(LS) \rightarrow \neg \text{term}(LS) \\
\neg ab(LS) \land \text{trans}(ls_{i}) \rightarrow \text{trans}(ls_{o}) \\
\vdots
\]

\(\neg \text{term}(LS)\) holds iff \(c\) spawns all of its vital processes/threads (\(\text{term}\) means "terminated"), and \(\text{trans}(c)\) is true iff (arbitrary) data is transmitted over connection \(c\). The models of SS and GS are analogous to LS.

Suppose that BD crashes. Then LS and SS will fail as well when they attempt to make a remote procedure call to the failed component BD: we assume that they throw software exceptions which are not handled, and so they terminate. Thus, the set of observations (\(\text{OBS}\)) contains the following literals: \(\{\text{term}(LS), \text{term}(SS), \neg \text{trans}(ls_{i}), \neg \text{trans}(ls_{o})\} \ldots\). There are no observations for the unobservable connections. The failure of BD is an independent failure, whereas LS and SS have failed dependently. However, in our scenario GS has not yet failed, as it makes only sporadic calls to BD.

We assume that the failure of BD cannot be directly determined due to the limited observability. Thus, we get the set of minimum diagnoses \(\Delta = \{\Delta\} \) with \(\Delta = \{ab(LS), ab(SS)\}\). Now we can try to repair the system. The authors of (Steinbauer & Wotawa 2005) propose to repair faults in a robot control system by restarting failed components. However, in our case the repair will fail if it is solely based on \(\Delta\): as BD is not repaired before, the two restarted components will immediately fail again when attempting to make a remote procedure call to BD.

It should be noted that the example system could also modeled in another way which considers the termination of LS and SS as a nominal reaction to a failure of BD, i.e., the termination of the former two components is not an abnormal behavior when it is caused by BD. In this case we would get 2 minimal diagnoses, namely \(\{BD\}\) and \(\{LS, SS\}\). Both are not satisfying, as they do not contain all components which are permanently "damaged" and which must be restarted.

The crucial point is that in systems with dependent failures the minimal diagnoses often do not contain all components which have failed. The focus on minimal diagnoses relies on the failure independency assumption. The usage of non-minimal diagnoses for repair is no solution for this problem, as their number may be very large, and they do not take failure dependencies into account. Furthermore, even the diagnosis \(\{ab(BD), ab(LS), ab(SS)\}\) is not satisfying, because it does not indicate the failure dependencies which influence the order in which the components must be repaired (BD must be repaired before LS and SS), and because it misleadingly indicates that LS and SS have failed due to internal errors (software bugs in this example).

In order to tackle these issues, we propose the concept of diagnosis environments (DEs). DEs are based on (not necessarily minimal) diagnoses; i.e., there is a set of DEs for each diagnosis, and the diagnoses are computed before creating their DEs (e.g., by Reiter’s Hitting Set algorithm). As described below, DEs embody the causal order of failures. For this purpose we augment the system model with a failure dependency graph (FDG) whose directed edges represent possible failure propagations, see Fig. 2. An edge from component \(c_i\) to \(c_j\) states that a failure of \(c_i\) may directly cause a failure of \(c_j\). Note that this does not mean that \(c_j\) must always fail when \(c_i\) fails.

For each component \(c\), a DE comprises one of the literals \(\neg ab(c), \text{indf}(c), \text{orf}(c), \text{df}(c)\), where \(\text{indf}(c)\) indicates an in-
dependent failure of \( c \), whereas \( df(c, e) \) states that \( c \) has dependently failed as a direct consequence of the failure of \( e \). The literal \( ab(c) \) does not occur in DEs, because any component which is abnormal has failed either independently or dependently. In other words, we regard the abnormality of a component as the consequence of its independent or dependent failure. For all components \( c \) which are abnormal in a diagnosis, either \( \text{indf}(c) \) or \( df(c, e) \) is in all DEs of this diagnosis. Moreover, taking the failure dependency graph into account, a DE may consider components as failed which are not abnormal in the diagnosis the DE refers to.

In this example, there are 7 possible DEs of \( \Delta = \{ ab(LS), ab(SS) \} \): \( \theta_1 = \{ \text{indf}(LS), \text{indf}(SS) \}, \theta_2 = \{ \text{indf}(BD), df(BD, LS), \text{indf}(SS) \}, \theta_3 = \{ \text{indf}(BD), df(BD, SS), \text{indf}(LS) \}, \theta_4 = \{ \text{indf}(BD), df(BD, LS), df(BD, SS) \}. \) The DEs \( \theta_5, \theta_6, \theta_7 \) are equal to \( \theta_2, \theta_3, \theta_4 \), respectively, with the only difference that they also contain \( df(BD, GS) \), because \( GS \) might also have failed due to the failure of \( BD \).

As multiple independent failures are unlikely in most applications, we focus on minimal DEs (MDEs) which have a minimal number of \( \text{indf}() \) literals. Thus, only \( \theta_2 \) and \( \theta_4 \) are minimal. However, the number of MDEs can still be large, and so we should seek to logically eliminate MDEs which are implausible.

We can achieve this by specifying abnormal behavior as well (Struss & Dressler 1989). For example, suppose only \( LS \) fails (i.e., it fails independently), and \( OBS = \{ \neg \text{trans}(ls_i), \text{trans}(ss_{s_j}) \} \). Then the set of minimal diagnoses contains, among others, \( \Delta = \{ ab(LS) \} \). So \( \theta = \{ \text{indf}(BD), df(BD, LS) \} \) may be a MDE of \( \Delta \). Hence, we improve our model and add

\[
\begin{align*}
ab(BD) \rightarrow \neg \text{trans}(ls_i) \land \neg \text{trans}(ss_{s_j}) \\
\text{trans}(ss_{s_j}) \rightarrow \text{trans}(ls_i) \\
\text{trans}(ls_i) \rightarrow \text{trans}(ss_{s_j})
\end{align*}
\]

Therefore, \( SD \). This captures our experience that, when \( BD \) fails, usually no more data is transmitted over its outputs. Now, as \( \text{trans}(ss_{s_j}) \) is observed, which implies \( \text{trans}(ls_i) \), the assumption \( ab(BD) \) is inconsistent with \( SD \cup OBS \).

Now we incorporate the following Dependency Mode Axioms (DMA)\(^2\):

\[
\begin{align*}
\text{indf}(c) \rightarrow ab(c) \\
\text{df}(c, c_j) \rightarrow ab(c_i) \land ab(c_j)
\end{align*}
\]

These axioms express that the abnormality of a component is a consequence of its independent or dependent failure. Consequently, the assumption \( \text{indf}(BD) \) is inconsistent with \( SD \cup DMA \cup OBS \), and there remains only one MDE for \( \Delta \), namely \( \{ \text{indf}(LS) \} \).

As a second possibility to reduce the number of MDEs we introduce the dependent failure description (DFD). It constrains the system behavior in case of dependent failures. We illustrate this by means of the first scenario with \( \Delta = \{ ab(LS), ab(SS) \} \). We got 2 MDEs: \( \theta_4 = \{ \text{indf}(BD), df(BD, LS), df(BD, SS) \} \) and \( \theta_7 = \{ \text{indf}(BD), df(BD, LS), df(BD, SS), df(BD, GS) \} \). As we assume that \( GS \) has not failed, our goal is to eliminate \( \theta_7 \). We know that \( GS \) terminates when it fails due to

\[
\text{df}(BD, GS) \rightarrow \text{term}(GS)
\]

to \( DFD \), and now \( \theta_7 \) is inconsistent with \( SD \cup DFD \cup DMA \cup OBS \), and so only one MDE remains, namely \( \theta_4 \).

**Diagnosis Environments (DEs)**

In (de Kleer, Mackworth, & Reiter 1992), a system is a tuple \((SD, COMP, OBS)\). We extend this definition:

**Definition 1 (System)** A system is a tuple \((SD, COMP, DFD, DF D, OBS)\). \( SD, DFD, DF D, \) and \( OBS \) are sets of first-order sentences. \( COMP \) is a finite set of components. \( DF D \) is a directed acyclic graph (DAG) whose nodes are components. There is exactly one node for every \( c \in COMP \).

As usual in Reiter’s framework, the system behavior is described in \( SD \) using the \( ab \) predicate. We do not restrict the usage of this predicate, i.e., it may also be used to specify abnormal behavior. Hence, the minimal diagnoses do not characterize all diagnoses (de Kleer, Mackworth, & Reiter 1992). We assume that \( SD \) does not take into account that components may fail dependently, and that neither the \( \text{indf} \) nor the \( df \) predicate occur in \( SD \cup OBS \).

All examples in the rest of the paper refer to the failure dependency graph which is depicted in Fig. 3. We use \( \text{parents}(c) \) to denote the set of all components which have an edge to \( c \) (e.g., \( \text{parents}(D) = \{ B, C \} \)). For simplicity, we assume that \( DF D \) is acyclic. Of course, in practice there may be components with mutual failure dependencies. However, this is beyond the scope of this paper.

![Figure 3: A failure dependency graph FDF.](image-url)
Definition 2 (dm(c) – Dependency Modes)
For all $c \in COMP$, $dm(c)$ denotes the set of literals \{\neg ab(c), indf(c)\} $\cup$ \{df(c, c) | $c_1$ is parents(c)\}.

Example 1 $dm(D) = \{\neg ab(D), indf(D), df(B, D), df(C, D)\}$, and $dm(F) = \{\neg ab(F), indf(F)\}$.

SD may contain descriptions of abnormal behavior, but DEs denote failed components by the $indf$ and $df$ predicates. Hence, we define axioms which state that the abnormality of a component $c$ is the consequence of its independent or dependent failure, and that a dependent failure of $c$ can only be caused by an abnormal component $c_i$:

Definition 3 (DMA – Dependency Mode Axioms)
DMA is a set comprising the following two axioms:

\begin{align*}
indf(c) & \rightarrow ab(c) \\
df(c_i, c_j) & \rightarrow ab(c_i) \wedge ab(c_j)
\end{align*}

We regard diagnoses and DEs as mode assignments:

Definition 4 (Mode Assignment (MA)) A mode assignment to a set of components $S \subseteq COMP$ is a set \$S \setminus \{m(c) \mid m(c) \in M(c)\}\$, $M(c)$ is a set of literals representing the possible modes of $c$. If $S = COMP$, the MA is complete, otherwise it is partial.

$M(c)$ depends on the context. We introduce a shortcut: Let $\omega$ denote a MA to a set of components $S$. We define $\Gamma(\omega) = \{c \mid \neg ab(c) \in \omega\}$, and $\Gamma(\omega) = S \setminus \Gamma(\omega)$. In other words, $\Gamma(\omega)$ contains all components considered as failed in $\omega$.

The following definition is analogous to Reiter’s. The diagnoses are based only on (SD, COMP, OBS):

Definition 5 (Diagnosis and Minimal Diagnosis) A diagnosis $\Delta$ for (SD, COMP, OBS) is a complete MA with $M(c) = \{\neg ab(c), ab(c)\}$ s.t. SD $\cup$ OBS $\cup$ $\Delta$ is consistent. $\Delta$ is (subset-)minimal iff there is no diagnosis $\Delta'$ s.t. $\Gamma(\Delta') \subset \Gamma(\Delta)$. Moreover, $\Delta$ is empty iff $\Gamma(\Delta) = \emptyset$.

Chains are required for the subsequent definition of DEs:

Definition 6 (Chain) Suppose $\omega$ is a MA with $M(c) = dm(c), |\omega| = k$ with $k \geq 2$, and $\Gamma(\omega) = \{c_1, \ldots, c_k\}$ (i.e., $\Gamma(\omega) = \emptyset$). Then $\omega$ is a chain from $c_1$ to $c_k$ iff the following holds: $df(c_{i-1}, c_i) \in \omega$ for $i = 2, \ldots, k$.

The notion of chain is similar to a path in graph theory. We use the notation $\{c_1 \rightarrow c_k\} \subseteq \omega$ to denote that an improper subset of a MA $\omega$ is a chain from $c_1$ to $c_k$. Basically, $\{c_1 \rightarrow c_k\} \subseteq \omega$ means that, according to $\omega$, the failure of $c_1$ has directly or indirectly caused a failure of $c_k$: directly if $k = 2$, indirectly otherwise. Obviously, $\{c_1, \ldots, c_k\}$ must be a path in FPG.

Example 2 Consider $\omega = \{indf(A), df(A, B), df(A, C), df(B, D)\}$. Several subsets of $\omega$ are chains: $\{indf(A), df(A, B), df(B, D)\}$, $\{indf(A), df(A, B)\}$, $\{df(A, B), df(B, D)\}$, $\{df(A, B), df(A, C)\}$.

Definition 7 (\$\Theta(\Delta)$ – Diagnosis Environments (DEs)) For a diagnosis $\Delta$, $\Theta(\Delta)$ denotes the set of all diagnosis environments of $\Delta$. Each $\theta \in \Theta(\Delta)$ is a complete MA with $M(c) = dm(c)$ for all $c \in COMP$ s.t. the following conditions hold:

1. $\Theta \cup \theta$ is consistent, $\Theta := SD \cup DFD \cup DMA \cup OBS$ for all $\theta \in \Theta(\Delta)$.

2. for all $c \in \Gamma(\Delta)$: $c \in \Gamma(\theta)$

3. for all $\theta$ with $indf(c) \in \theta$: either $c \in \Gamma(\Delta)$ or there is a $\theta' \in \Gamma(\Delta)$ s.t. $c \cap c' \subseteq \theta$.

$\Theta$ is a useful shortcut which is used throughout the rest of this paper.

Note that this definition does not require $\Delta$ to be minimal. Moreover, it does not define minimal DEs (MDEs); MDEs will be defined below.

Item 2 of Def. 7 states that all components $c$ which are abnormal in $\Delta$ are considered as failed in $\theta$ as well, i.e., either $indf(c) \in \theta$ or $\forall (\theta, c) \in \theta$. Item 3 restricts which components may be assumed as independently failed: $indf(c)$ can only be in $\theta$ if either $ab(c) \in \Delta$ or $c$ may have, directly or indirectly, caused a failure of any $\theta'$ with $ab(\theta') \in \theta$.

Def. 7 implicitly states which $df$ modes may be in a DE:

Theorem 1 If $\theta$ is a DE, then for a component $c_i$ with $df(c_k, c_i) \in \theta$ there is exactly one component $c$ with $indf(c) \in \theta$ and $\{c \cap c_i\}$ $\subseteq \theta$.

Proof. Suppose $DMA \cup \{df(c_k, c_i)\} = \emptyset$. Let $\Delta$ be a diagnosis of $\Theta$. Then the following MAs are DEs of $\Delta$, provided they are consistent with $T$ (Def. 7): $\{indf(B), indf(C)\}$, $\{indf(A), df(A, B), indf(C)\}$, $\{indf(A), df(A, B), df(A, C)\}$, $\{df(A, B), df(D, C)\}$, $\{indf(A), df(A, B), df(D, C)\}$, $\{indf(A), df(A, B), df(D, C)\}$, etc. Note that neither $P$ nor $G$ are failed in any DE of $\Delta$.

Example 3 Suppose $\Delta = \{ab(B), ab(C)\}$ is a diagnosis. Then the following MAs are DEs of $\Delta$, provided $\Gamma(\Delta)$ contains a chain from $c_1$ to $c_k$: $\{indf(A), df(A, B), df(A, C), df(B, D)\}$, $\{indf(A), df(A, B), df(A, C), df(B, D)\}$, $\{indf(A), df(A, B), df(A, C), df(B, D)\}$, $\{indf(A), df(A, B), df(A, C)\}$, $\{indf(A), df(A, B), df(A, C)\}$.

Example 4 Suppose $\Delta = \{ab(A), ab(F)\}$. Examples for DEs are: $\{indf(A), df(A, B), df(A, C), df(B, D), df(A, C), df(B, D)\}$, $\{indf(A), df(A, B), df(A, C), df(B, D)\}$, $\{indf(A), df(A, B), df(A, C), df(B, D)\}$, etc.

Example 5 Suppose $\Delta = \{ab(D)\}$. Then $\{indf(A), df(A, B), df(A, C), df(C, D)\}$ and $\{indf(E), df(E, C), df(C, D)\}$ are among the possible DEs. Again, $P$ and $G$ are not abnormal in all DEs of $\Delta$.

We introduce another shortcut: for a MA $\omega$, the $\sigma$ function substitutes some modes in $\omega$ and returns the new MA. For example, $\sigma(\omega, ab/indf)$ substitutes every literal $ab(c)$ in $\omega$ by $indf(c)$, the other modes are not changed. Furthermore, $\sigma(\omega, ab/indf)$ substitutes only the mode of a specific component $X$.

Intuitively, we expect to obtain a DE of a diagnosis $\Delta$ by substituting every literal $ab(c)$ in $\Delta$ by $indf(c)$. Our view is that, if a component behaves abnormally, then it is always possible that an internal error is the cause. In order to achieve this we impose formal restrictions on FDF:

Definition 8 (Restrictions on FDF) The predicates $ab$ and $indf$ must not occur in FDF. Furthermore, FDF must not interfere with SD; i.e., it must be guaranteed that for any diagnosis $\Delta$ also $SD \cup FDF \cup OBS \cup \Delta$ is consistent.
Since we also assume that the \textit{indf} predicate is not used in $SD\cup OBS$, this predicate only occurs in $DMA$. Now we can provide a theorem which complies with our intuitions:

\textbf{Theorem 2} If $\Delta$ is a diagnosis then $\theta = \sigma(\Delta, ab/\textit{indf})$ is a DE of $\Delta$.

\textit{Proof.} Due to Def. 5 and Def. 8, $SD\cup DF\cup OBS\cup \Delta$ is consistent. Because $\neg ab(c) \in \theta$ iff $\neg ab(c) \in \Delta$ and the \textit{indf} predicate does not occur in $SD\cup DF\cup OBS$, we know that $SD\cup DF\cup OBS\cup \theta$ is consistent. Finally, for all literals $ab(c)\notin DMA\cup \theta \models ab(c)$ [i.e., $\textit{indf}(c) \in \theta$, also $ab(c) \in \Delta$. Hence, $SD\cup DF\cup DMA\cup OBS\cup \theta$ is consistent. The rest of the proof is trivial. 

Therefore, each diagnosis has at least one DE. The following two corollaries follow from Def. 7 and from Theorem 2:

\textbf{Corollary 1} Let $\Delta$ be a diagnosis. If $FDG$ has no edges, then $\Theta(\Delta) = \{\theta\} = \{\sigma(\Delta, ab/\textit{indf})\}$.

\textbf{Corollary 2} If $\Delta$ is an empty diagnosis, then $\Theta(\Delta) = \{\}$. 

An important consequence of the definition of DEs is that each $\theta \in \Theta(\Delta)$ corresponds to a diagnosis $\Delta'$ with $\Gamma(\Delta') \supseteq \Gamma(\Delta)$:

\textbf{Theorem 3} If $\Delta$ is a diagnosis, then for all $\theta \in \Theta(\Delta)$ the following holds:

\[ \Delta' = \left[ \bigcup_{c \in \Gamma(\theta)} \neg ab(c) \right] \cup \left[ \bigcup_{c \in \Gamma(\theta)} \{ab(c)\} \right] \text{ is also a diagnosis, and } \Gamma(\Delta') \supseteq \Gamma(\Delta). \]

\textit{Proof.} For all $c \in \Gamma(\theta)$: $DMA\cup \theta \models ab(c)$. Now it follows from Def. 7 that $\Upsilon \cup \theta \cup \{ab(c)\} \in \Gamma(\theta)$ is consistent, and so $SD\cup OBS\cup \Delta'$ is consistent, too. Hence, $\Delta'$ is a diagnosis. Moreover, from $c \in \Gamma(\Delta)$ it follows that $c \in \Gamma(\theta)$ and $\Gamma(\theta) = \Gamma(\Delta')$. Hence, $\Gamma(\Delta') \supseteq \Gamma(\Delta)$. 

Now we extend the notion of DE to a set of diagnoses:

\textbf{Definition 9 (\textit{\Theta}(\mathcal{D}) \text{ - DEs of a set of diagnoses}) If \mathcal{D} is a set of diagnoses, then $\Theta(\mathcal{D}) = \bigcup_{\Delta \in \mathcal{D}} \Theta(\Delta)$.}

It should be noted that for two different diagnoses $\Delta_1$ and $\Delta_2$ it is possible that $\Theta(\Delta_1) \cap \Theta(\Delta_2) \neq \emptyset$.

\textbf{Example 6} Suppose $\Delta_1 = \{ab(A)\}$ and $\Delta_2 = \{ab(B)\}$. Then $\theta = \{\textit{indf}(A), df(A, B)\}$ may be a DE of both $\Delta_1$ and $\Delta_2$, i.e., $\theta \in \Theta(\Delta_1)$ and $\theta \in \Theta(\Delta_2)$.

As the number of DEs can be daunting, we need a notion of \textit{minimality}: minimal DEs have a minimal number of \textit{indf} literals and they are, unlike DEs, only defined for minimal diagnoses. For the following definition we introduce a shortcut: for any MA $\omega$, let $\#(\omega)$ denote the number of \textit{indf} literals in $\omega$.

\textbf{Definition 10 (Minimal DEs (MDEs)) If \mathcal{D} is a set of minimal diagnoses, then $\theta \in \Theta(\mathcal{D})$ is minimal in $\Theta(\mathcal{D})$ iff the following holds for all $\theta' \in \Theta(\mathcal{D})$: $\#(\theta) \geq \#(\theta')$.}

\textbf{Example 7} Suppose $\mathcal{D} = \{\Delta_1, \Delta_2\}$ with $\Delta_1 = \{ab(B), ab(C)\}$ and $\Delta_2 = \{ab(F)\}$, and $SD\cup OBS\cup \{ab(A)\}$ is inconsistent. Then $\theta = \{\textit{indf}(A)\}$ is inconsistent, too, and so $\#(\theta) = 2$ for all $\theta \in \Theta(\Delta_1)$, because all $\theta$ contain $\textit{indf}(B)$ and either $\textit{indf}(C)$ or $\textit{indf}(E)$. Therefore, all DEs of $\Delta_1$ are not minimal in $\Theta(\mathcal{D})$, because $\{\textit{indf}(F)\} \notin \Theta(\mathcal{D})$.

This example is very important, as it shows that there may be minimal diagnoses which do not contribute to the set of all MDEs of a system. Our algorithm which computes all MDEs for a system will benefit from this insight by computing only those diagnoses whose DEs may be minimal.

Clearly, if there is an empty diagnosis $\Delta$ (i.e., all components are not abnormal in $\Delta$), then $\Delta$ is also a DE (see Corollary 2), and it is the sole MDE of the system. The following proposition states the minimum and maximum number of \textit{indf} modes for all DEs of a non-empty diagnosis:

\textbf{Proposition 1} For a non-empty diagnosis $\Delta$ the following holds for all $\theta \in \Theta(\Delta)$:

\[ 1 \leq \#(\theta) \leq \#(\sigma(\Delta, ab/\textit{indf})) = |\Gamma(\Delta)| \]

\textit{Proof.} From Def. 7 and Theorem 1 it follows that each DE of a non-empty diagnosis must contain at least one \textit{indf} literal. Furthermore, suppose $\#(\theta) > |\Gamma(\Delta)|$. Then, due to Item 3 of Def. 7, there must be components $c, c'$ and $c''$ ($c' \neq c''$), s.t. the following holds: $c \in \Gamma(\Delta)$, $\textit{indf}(c') \in \theta$, $\textit{indf}(c'') \notin \theta$, $c' \wedge c \subseteq \theta$ and $c'' \wedge c \subseteq \theta$. However, this conflicts with Theorem 1. 

\textbf{Simple Algorithm}

We first introduce a naive algorithm which computes all MDEs for a system, and describe important optimizations in the following section. We use $\mathcal{D}$ to denote the set of all minimal diagnoses, and $\Theta(\mathcal{D})$ is the set of all MDEs of the system.

The algorithm first computes the minimal diagnoses and then creates a \textit{diagnosis environment graph} (DEG). The DEG is a forest, i.e., it comprises disjoint trees, where each tree relates to a minimal diagnosis. The naive algorithm requires all minimal diagnoses, hence for $|\mathcal{D}| = m$ the DEG contains $m$ trees $T_1, \ldots, T_m$. Each node $n$ of one of the trees represents a MA, denoted by $n.\omega$, which may be a DE. More precisely, $n.\omega$ is a DE (but not necessarily a MDE) iff $\Upsilon \cup n.\omega$ is consistent.

Figures 4 and 5 depict the trees for the diagnoses $\Delta_1 = \{ab(B), ab(C)\}$ and $\Delta_2 = \{ab(A), ab(B)\}$, respectively. For any diagnosis $\Delta$, the corresponding tree $T_\Delta$ has a root node $n$ with $n.\omega = \sigma(\Delta, ab/\textit{indf})$. A node has the type $\alpha$ or $\beta$. The root node is an $\alpha$-node. The children of an $\alpha$-node may be of type $\alpha$ or $\beta$, whereas all children of a $\beta$-node are $\beta$-nodes as well. In Fig. 4 and 5, the edges to children of type $\alpha$ are depicted by solid lines, whereas the edges to $\beta$-children are dotted.

Algorithms 1 and 2 describe how a node is expanded. Remember that $\textit{parents}(c)$ denotes the parents of $c$ in $FDG$. Basically, when creating an $\alpha$-child of $n$ (i.e., a child of type $\alpha$), a mode $\textit{indf}(c) \in n.\omega$ is substituted by $df(c_i, c)$ and the mode of $c_i$ may be set to $\textit{indf}(c_i)$, whereas a $\beta$-child of $n$ is generated by replacing a mode $\neg ab(c)$ by $df(c_i, c)$.

It is important to note that these node expansion algorithms may create a new node $n_i$, although the DEG already contains a node $n_i$ with $n_i.\omega = n.\omega$. The nodes $n_i$ and $n_j$ may belong either to the same tree or to different trees. In both cases, $n_i$ may be discarded (i.e., not added to the DEG), because the MAs of all descendant nodes of $n_i$ (children, grandchildren, great-grandchildren etc.) are redundant. This is trivial to see if $n_i$ and $n_j$ have the same types, but it can be shown that this also holds if these nodes have different types (we omit the proof).
Algorithm 1: Create $\alpha$-children of node $n$

Input: An $\alpha$-node $n$

Output: A modified DEG containing new $\alpha$-nodes

1. For all $c$ with $\text{indf}(c) \in n.\omega$:
   1. Create a new $\alpha$-node $n'$
      1. $n'.\omega := \sigma(n.\omega, \text{indf}(c)/\text{df}(c_1, c))$
      2. If $\neg\text{ab}(c_1) \in n'.\omega$:
         1. $n'.\omega := \sigma(n'.\omega, \neg\text{ab}(c_1)/\text{indf}(c_1))$
      3. Add $n'$ to DEG (as child of $n$)

Algorithm 2: Create $\beta$-children of node $n$

Input: An $\alpha$- or $\beta$-node $n$

Output: A modified DEG containing new $\beta$-nodes

1. For all $c$ with $\neg\text{ab}(c) \in n.\omega$:
   1. For all $c_i \in \text{parents}(c)$:
      1. If $c_i \in \Gamma(n.\omega)$:
         1. Create a new $\beta$-node $n'$
            1. $n'.\omega := \sigma(n.\omega, \neg\text{ab}(c)/\text{df}(c_1, c))$
         2. Add $n'$ to DEG (as child of $n$)

Therefore, the algorithm should ensure that only unique MAs are generated. The simplest possibility is to compare each new MA with the already created nodes and to add the new node to the DEG only if its MA is unique.

Example 8 The MAs in Fig. 4 and 5 are unique within their trees. For example, the node $X_4$ could have an $\alpha$-child with the MA $\{\text{indf}(A), \text{df}(A, B), \text{df}(A, C)\}$, but this MA would be equal to that of $X_3$. However, there are MAs which are depicted in both trees: $X_3$ and its descendents are equivalent to $X_3$ and its descendents (and vice versa). Thus, if $T_{\Delta_x}$ is created after $T_{\Delta_x}$, then the nodes $X_3$ and $X_7$ can be discarded, and $T_{\Delta_y}$ contains only 4 nodes, namely $Y_1$, $Y_2$, $Y_5$, and $Y_{10}$.

A naive algorithm which computes all MDEs of a system works as follows. First, all minimal diagnoses of the system are generated, e.g. by using Reiter’s Hitting Set algorithm. If there is an empty diagnosis, then this diagnosis is the sole MDE of the system (see Corollary 2), and the algorithm is finished. Otherwise, the entire DEG is created (with unique mode assignments). Afterwards, the algorithm performs consistency checks for all nodes $n$ with $\#(n.\omega) = 1$, i.e., the consistency of $\bigcup n.\omega$ is checked. If consistent MAs are found, then they compose $\Theta(\mathbb{D})$. Otherwise, the consistency of nodes with 2 indf modes must be checked as well, etc. Note that it is guaranteed that the algorithm terminates, because the root nodes are DEs (see Theorem 2).

Improved Algorithm

A severe drawback of the naive algorithm is that it requires the generation of all minimal diagnoses and that it always creates the entire DEG. The improved algorithm seeks to cut the search space by creating only a part of the DEG. It initially assumes that there are MDEs with only one indf mode (only if there is no empty diagnosis, of course), and so it does not expand a node $n$ if it has more than one indf mode and if it is clear that the same would hold for all descendants of $n$. In addition, if $n$ is a root node, which directly stems from a minimal diagnosis $\Delta$, then we know that all DEs of $\Delta$ contain more than one indf mode, and so it is not even necessary to generate this diagnosis. In other words, when the diagnosis algorithm encounters the MA $\Delta$, 

Figure 4: The DEG tree $T_{\Delta_x}$ for $\Delta_x = \{ab(B), ab(C)\}$ containing only unique MAs. Nodes are labelled with $X_i$.

Figure 5: The DEG tree $T_{\Delta_y}$ for $\Delta_y = \{ab(A), ab(B)\}$.
then it does not need to check if $\Delta$ is a diagnosis. E.g., when
Reiter’s Hitting Set algorithm is used, the consistency check
of $SD \cup OBS \cup \Delta$ and the expansion of the corresponding
node in the HS-DAG can be omitted (Reiter 1987).

If there is no MDE with one $indf$ mode, then the algo-
rithms searches for MDEs with two $indf$ modes, etc. So the
generation of minimal diagnoses and the expansion of the
DEG is an incremental process.

For this purpose, we introduce a function $\#_\geq(n)$ which
returns a lower boundary for the number of $indf$ modes of $n$
and of all of its descendants. Formally, this function returns a
number $\zeta$ s.t. $\#(n.\omega) \geq \zeta$ and $\#(n'.\omega) \geq \zeta$ for all
descendants $n'$ of $n$. We use $\#_\geq(\Delta)$ as a shortcut for $\#_\geq(n)$
where $n$ is a root node with $n.\omega = \sigma(\Delta, ab/indf)$.

Trivially, $\#_\geq(n)$ could always return 1, but we seek a
better estimation. If $n$ is a $\beta$-node, then $\#(n.\omega) = \#(n'.\omega)$
for all descendants $n'$ of $n$. Thus, $\#_\geq(n) = \#(n.\omega)$. For
$\alpha$-nodes, an estimation can be obtained as follows: Suppose
$n$ has 2 $indf$ modes, namely $indf(c_1)$ and $indf(c_2)$. Then
$\#_\geq(n) = 1$ if there is a directed path in FDG from $c_1$
(or vice versa) or if there is a component $c_0$ s.t. there are
directed paths in FDG from $c_0$ to both $c_1$ and $c_2$. This idea
can be generalized to $\#(n.\omega) = k$ with $k \geq 2$. We do not
show this here due to space reasons.

Example 9 Consider node $X_1$ in Fig. 4. As component
$A$ has paths in FDG to both $B$ and $C$, there may be de-
cendants of $X_1$ with only one $indf$ mode, namely $indf(A)$,
and so $\#_\geq(X_1) = 1$. However, $\#_\geq(X_{11}) = 2$, as none
of the conditions above hold. Clearly, all descendants of
$X_{11}$, which are of type $\alpha$ or $\beta$, contain $indf(E)$ and ei-
ther $indf(B)$ or $indf(A)$. In Fig. 5, $Y_3$ is a $\beta$-node, and
so $\#_\geq(Y_3) = \#(Y_3.\omega) = 1.$

Note that the estimation presented above can be effi-
ciently implemented by using two pre-compiled $|COMP| \times
|COMP|$ matrices in which each entry $(i, j)$ is a boolean value
indicating (1) if $c_i$ has a path to $c_j$ in FDG and (2) if
there is a component $c_0$ which has paths to both $c_1$ and $c_j$.
These matrices can be computed in $O(|COMP|^{13}).$

Algorithm 3 outlines an improved algorithm which uti-
lizes $\#_\geq(n)$. The generation of minimal diagnoses and the
expansion of the DEG is incremental.

Algorithm 3: Computes all MDEs of a system
Input: A system
Output: $\Theta(\Delta)$
(1) If there is an empty diagnosis $\Delta$, return $\Theta(\Delta) := \{\Delta\}$
(2) For $\nu := 1, \ldots, |COMP|$
(3) Generate all minimal diagnoses $\Delta$ with $\#_\geq(\Delta) = \nu$;
add them as root nodes to the DEG.
(4) Repeat
(5) Select an unexpanded $\alpha$-node $n$ s.t. $\#_\geq(n) = \nu$
(6) Create $\alpha$-children of $n$ (Alg. 2)
(7) until no more nodes are created
(8) For all $n$ with $\#(n.\omega) = \nu$, create all descendants of
$\beta$ (i.e., apply Alg. 2 repeatedly)
(9) $\Theta(\Delta) := \{n.\omega | \#(n.\omega) = \nu \text{ and } \Upsilon \cup n.\omega \not\subseteq \Delta\}$
(10) If $\Theta(\Delta) \neq \emptyset$, finish algorithm, return $\Theta(\Delta)$

Example 10 Suppose we have 2 minimal diagnoses,
$\Delta_x = \{ab(B), ab(C)\}$ and $\Delta_z = \{ab(B), ab(E)\}$. The
DEG tree for $\Delta_x$ is depicted in Fig. 4, the tree for $\Delta_z$ is not
depicted. In the first iteration ($\nu = 1$), only $\Delta_z$ is gener-
ated, as $\#_\geq(\Delta_z) = 2$. The following $\alpha$-nodes are created:
$X_1, X_2, X_3, X_8$, and $X_{11}$. However, $X_{11}$ is not expanded
because $\#_\geq(X_{11}) = 2$. Then the $\beta$-nodes $X_4$ and $X_5$ are
generated (line 8 in Alg. 3). Finally, consistency checks are
performed for $X_3, X_4$, and $X_5$ (line 9). If at least one of
these MAs is consistent with $\Upsilon$, then the algorithm termi-
nates. Otherwise, $\Delta_z$ is generated as well and added to
the DEG as root node of a second tree. In the second iteration,
all remaining nodes of the DEG are created.

A further improvement of the algorithm can be achieved
by utilizing conflict sets. The following definition is similar
to those in (Reiter 1987) and (de Kleer & Williams 1989):

Definition 11 (Conflict Set) A conflict set $\gamma$ is a partial MA
with $M(e) = dne(c) \cup \{ab(c)\}$ s.t. $\Upsilon \cup \gamma$ is inconsistent.

I.e., a conflict set may contain the literals $\neg ab(-), indf(-),
df(-)$, and also $ab(-)$ because $SD$ may contain descriptions
of abnormal behavior. We suppose that the reasoner which
performs the consistency checks of $\Upsilon \cup \omega$ returns a conflict
set, denoted by $n.\gamma$, if an inconsistency is detected. Note
that $n.\gamma \subseteq n.\omega$.

The crucial insight is that it is often possible to infer from
$n.\gamma$ that the MAs of all descendants of $n$ must be inconsist-
ent with $\Upsilon$ as well. Let $n.\omega \subseteq n.\omega$ denote a partial MA s.t.
$n.\omega \subseteq n.\omega$ for all descendants $n'$ of $n$. In other words, $n.\omega$
contains all those modes of $n.\omega$ which are also contained in
all descendants of $n$. Then, if $n.\gamma \subseteq n.\omega$, all descendants
must be inconsistent with $\Upsilon$, and we can prune the tree.

Therefore, Alg. 3 can be improved by performing the consis-
tency checks for nodes $n$ with $\#(n.\omega) = \nu$ already during
the expansion of the DEG, and by omitting the expansion of
a node $n$ when it is clear that all descendants of $n$ would
be inconsistent with $\Upsilon$. $n.\omega$ can be easily derived from
the structure of the DEG and from the failure dependency graph.

Example 11 Suppose the consistency check for $X_3$ (Fig. 4)
yields the conflict set $X_3.\gamma = \{indf(A), df(A, C),
\neg ab(E), \neg ab(F), \neg ab(G)\}$. As $X_3.\omega = \{indf(A), df(A, C),
\neg ab(E), \neg ab(F), \neg ab(G)\}$, $X_3$ is not expanded.

We also observe that the number of required consistency checks
can be reduced by preserving computed conflict sets in a
data structure (similar to the nogood database in an
ATMS, see (de Kleer 1986)). Before performing a consistency
check for a certain MA, we can search in this data
structure for a conflict set which refutes this MA.

Example 12 Suppose, as in the previous example, that
$X_2.\gamma \subseteq X_2.\omega$, $X_8$ (and its descendants) must be inconsistent.

Finally, we present important properties of conflict sets:

Proposition 2 If $\gamma$ is a conflict set, then $\sigma(\gamma, ab/indf)$ and
$\sigma(\gamma, indf/ab)$ are conflict sets, too.

Proof. Suppose $ab(c) \in \gamma$. As $DMA \cup \{indf(c)\} =
\{ab(c), \Upsilon \cup \gamma \} \cup \{ab(c)\}$ is inconsistent as well. Moreover,
the $indf$ predicate occurs only in $DMA$ (see Def. 8).
$\sigma(\gamma, indf/ab)$ must also be inconsistent.

Proposition 3 If $\gamma$ is a conflict set and $ab(c) \in \gamma$, then
$\sigma(\gamma, ab(c)/df(-, c))$ is a conflict set, too.
Discussion and Related Research

We proposed the concept of MDEs in order to tackle the issue of dependent failures in consistency-based diagnosis. We motivated our work using an example from the control system of an autonomous robot. We provided a formalization of our approach, and we discussed consequences of our definitions. Furthermore, we provided an algorithm which computes all MDEs of a system.

In case of dependent failures, the minimal diagnoses often do not contain all components which have failed. Thus, each MDE corresponds to a (in general not minimal) diagnosis, and MDEs reflect the causal order of failures. The latter is important as the order of failure often indicates the order in which components must be repaired.

The authors of (de Kleer, Mackworth, & Reiter 1992) propose to compute kernel diagnoses instead of minimal diagnoses. However, we hold the view that in many applications multiple failures are only likely if they are dependent failures. Therefore, we explicitly model these dependencies by means of the failure dependency graph, which captures all possible failure dependencies. This graph is used to focus the reasoning.

There is no general answer how a modeller can create the graph. In some applications it might be possible to compute the graph dynamically, based on a domain-specific model and on current observations. In this paper we assume that the modeller knows the possible failure dependencies.

In many cases the number of MDEs will be much smaller than the number of kernel diagnoses. In some cases, there may be even fewer MDEs than minimal diagnoses, because MDEs are cardinality minimal wrt inf modes. However, the number of MDEs can still be daunting. Apart from the focusing strategies we outlined above, the description of abnormal behavior (in SD) and of behavior in case of dependent failures (in DF D) can significantly reduce the number of MDEs. For example, one could adopt the view that a component must be correct if it manifests only nominal behavior in the given observations (Raiman 1989).

The complexity of the computation of MDEs depends on different factors. The density of the failure dependency graph and the length of its paths have a large impact. In general, our approach may be intractable if the cascade of dependent failures is not locally confined in a large system, i.e., if the dependent failures propagate throughout the entire system. Another issue is the fact that dependent failures can lead to minimal diagnoses with a high cardinality. Our algorithm addresses this problem by (incrementally) generating only those minimal diagnoses whose DEs may be minimal.

The example at the beginning of this paper is from an autonomous robot. (Steinbauer & Wotawa 2005) and (Weber & Wotawa 2006) deal with diagnosis in autonomous robots; however, dependent failures are not considered.

An empirical evaluation of the performance of our algorithm will be part of our future work. Another open issue is the combination of our approach with behavioral modes (de Kleer & Williams 1989). Furthermore, future research should deal with the application of probabilistic focusing techniques.

References


Raiman, O. 1989. Diagnosis as trial - the alibi principle. In International Model-Based Diagnosis Workshop.


