

# Managing diagnosis processes with interactive decompositions

Quang-Huy GIAP, Stephane PLOIX, and Jean-Marie FLAUS

**Abstract** In the scientific literature, it is generally assumed that models can be completely established before the diagnosis analysis. However, in the actual maintenance problems, such models appear difficult to be reached in one step. It is indeed difficult to formalize a whole complex system. Usually, understanding, modelling and diagnosis are interactive processes where systems are partially depicted and some parts are refined step by step. Therefore, a diagnosis analysis that manages different abstraction level and partly modelled components would be relevant to actual needs. This paper proposes a diagnosis tool managing different modeling abstraction levels and partly depicted systems. The main idea is to exploit the information coming from the links between the different levels of abstraction, to infer non modelled parts and to use the redundancies existing in the knowledge of the system (e.g. fault propagation or shared resources) to improve diagnosis results.

## 1 Introduction

In the diagnosis community, abstraction has been presented as a promising technique to reduce the computational cost of model-based diagnosis [10, 1, 2, 4]. First, abstract procedure tends to aggregate items to describe the system at different levels of abstraction with different level of details (structural and behavioral). It is called

---

Quang-Huy GIAP

Laboratoire G-SCOP, 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 - France, e-mail: quang-huy.giap@g-scop.inpg.fr

Stephane PLOIX

Laboratoire G-SCOP, 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 - France, e-mail: stephane.ploix@g-scop.inpg.fr

Jean-Marie FLAUS

Laboratoire G-SCOP, 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 - France, e-mail: jean-marie.flaus@g-scop.inpg.fr

*bottom-up method* because it begins by the terminal level and stops in the most abstract level. Then, algorithms, which are based on Mozetic's approach, are proposed to solve the problem. Contrary to *bottom-up method*, a *top-down method* is proposed. The important point of our purpose is to use abstraction to fit the actual diagnosis process in the context of human machine cooperation. Indeed, it has been observed that a human facing a diagnosis problem starts by global observations taking into account abstract items then, step by step, the description of the problem is refined in a direction that depends on the observed symptoms. It is therefore an important matter to manage these consecutive refinements in such a top-down method.

The paper is organized as follows. In section 2, the underlying modelling concepts are introduced. Main concepts of semantic theory of abstraction are recalled in order to be able to model all the aspects of a diagnosis problem. Section 3 presents the basic principles used to exploit the multi-level models. Then, an algorithm is proposed in section 4, which is inspired from the HS-Tree algorithm [15, 9]. In section 5, an example is taken to illustrate the problem.

## 2 Problem statement

Physical system modelling is related to different point of views [3]: structural aspect (what is in a system and how its components are connected), behavioral aspect (how does each element work), functional aspect (what does each element do) and teleological aspect (what is each element for). In the literature about theory of abstraction, [4] used theory of abstraction to manage behavior together with different levels of structural abstraction [11] in diagnosis. It allows reducing the complexity of reasoning. In order to construct a multi-level abstraction model that can be used in diagnosis analysis based on DX approach [15], the only three first levels are considered. The teleological level, which is useful to discover the causes of faults, is out of the scope of diagnosis reasoning. In this paper, the term *item* is preferred instead of *component* because in actual applications different types of elements may be encountered such as functions, operations, components. Moreover, in a multi-abstraction level context, super-functions and a super-components use to appear.

### 2.1 Phenomenological modelling

In a physical system, a phenomenon is a directly observable element of information about the state of a system. It is usually modelled by physical variables. The set of all the possible values of a variable  $v$ , which allows depicting a phenomenon is called the value domain of  $v$ , denoted by  $dom(v)$ . According to the related value domain, a variable may be qualitative, numeric discrete or continuous. On the other hand, variables may be divided into two groups according to structural description. The first one is called *internal variable*, it is used to depict an *item* and it is not

shared with any other item. The second one is called *external variable* because it is shared by different items. *External variables* represent the exchange of information or material flows between items, or between items and the external environment.

## 2.2 Behavioral and functional modelling

Behaviors and functions rely on physical variables and possibly on observed values. They may be represented by constraints defined on a space generated by the involved variables, called domain space of variable. In Artificial Intelligence community, a constraint represents the relationship between variables. It enumerates either the continuous relationship between variables or, in a discrete context, the possible tuples of values from the domain space of variable that variables may take. As defined in [12], let  $V = \{v_1, v_2, \dots, v_n\}$  be a tuple of variables,  $\kappa$  is a constraint over  $V$  if  $\kappa \subset \text{dom}(V)$  where  $\text{dom}(V) = \text{dom}(V_1) \times \dots \times \text{dom}(V_n)$ . The set  $V$  is then denoted  $V = \text{var}(\kappa)$ .

The *behavior* of an item is modelled by constraints characterizing the set of possible values of involved variables. It will be called a zone of the domain space of variable. Generally, just some parts of the domain space of variable are physically possible. The zone of physically possible values is named *possible zone*. It is denoted  $PZ$ . Of course, this zone contains all the zones that corresponds to specific behaviors [8]. The complementary part of *possible zones* in domain space of variable is called *physically impossible zone*, denoted  $IZ$  for *Impossible Zone*. It means that there is no way to reach this zone, even in faulty behavior.

The *behavioral mode* of an item is modelled by one or more constraints. In the *possible zone*, an item may be depicted by one correct mode and by different incorrect modes. The mode corresponding to normal behavior or normal operation, denoted  $ok(item)$  (normal mode), is modelled by a constraint that defines the zone of normal behavior:  $NZ(item)$ . A fault or failure mode, denoted  $fm_i(item)$ , is also modelled by one or more constraints corresponding to a zone of abnormal behavior denoted  $FZ_i(item)$  for *Fault Zone*. Another zone exists, it is named *complementary zone* ( $CZ$ ). It refers to the part of the possible zone that do not correspond to normal mode nor to modelled fault mode:  $CZ = PZ \setminus (NZ \cup \bigcup_i FZ_i)$ . It corresponds to a virtual complementary fault mode denoted  $cfm(item)$ . The  $cfm$  mode is considered as an unmodelled faulty mode because it does not correspond to any constraint and its zone  $CZ(item)$  does not belong to the normal zone. The different zones  $NZ(item)$  and  $FZ_i(item)$  may be intersected. For example, the behavior in a given fault mode may be assimilated sometimes to a normal mode and to other fault modes. However, there are no intersections between a complementary zone specifying a complementary fault mode and any other zone because a complementary zone is defined as the complementary part of all other zones in the possible zone  $PZ$  (see figure 1).

The set of behavioral modes of an item may be written:

$$Modes(item_i) = \{ok, [fm_1, \dots, fm_n], cfm\} \quad (1)$$

An item is called *non-modelled* if there is no available constraint that represents any of its modes. However, it is convenient to assume the existence of 2 modes *ok* and *cfm* for such an item that can be depicted as a part of another item. It is discussed in the next subsection.

### 3 Abstraction in model-based diagnosis

This section depicts how main phenomena can be formalized in a multi-abstraction level context.

#### 3.1 Formalizing abstraction

Let's consider *behavioral abstraction*. As mentioned before, an item is either a function or a physical resource. The hierarchical decomposition of a system is generally begun by the global function of the system i.e. the most abstract item. Then, this item may be decomposed into sub-items that may be sub-functions, sub-components, . . . In other words, an expected behavioral mode of an item is achieved by its sub-items. In order to formalize hierarchical relations between items, let's introduce the notion of m-proposition.

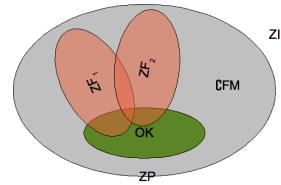
**Definition 1.** (m-proposition) A logical proposition where symbols are modes of items, which can be expressed by a conjunctive normal form, is called a m-proposition. If  $\mathcal{P}(mode_1, \dots, mode_n)$  is a m-proposition, the support  $\mathcal{P}$  is defined by  $Modes(\mathcal{P}) = \{mode_1, \dots, mode_n\}$ .

For example,  $(mode_1 \rightarrow mode_2) \wedge mode_3$ , with  $\neg mode_1 = mode_4 \vee mode_5$ , is a m-proposition because it can be rewritten as:  $(mode_2 \vee mode_4 \vee mode_5) \wedge mode_3$ .

**Definition 2.** (monomial of m-proposition) A monomial in a m-proposition is one of the disjunctive proposition appearing in the equivalent conjunctive normal form.

For instance, in the previous example,  $mode_2 \vee mode_4 \vee mode_5$  and  $mode_3$  are the monomials of the m-proposition.

**Fig. 1** Representation of the different zones describing the behavioral/operating modes



Then, based on a hierarchical decomposition, the concept of sub-item may be defined.

**Definition 3.** (sub and super-item) An item  $I_2$  is a sub-item of an item  $I_1$  if it exists at least one mode  $mode_i(I_1)$  of the item  $I_1$ , one mode  $mode_j(I_2)$  of the item  $I_2$  and a m-proposition  $\mathcal{P}$  such as  $mode_i(I_1) \rightarrow \mathcal{P}$  with  $mode_j(I_2) \in Modes(\mathcal{P})$ . If  $I_2$  is a sub-item of  $I_1$  then  $I_1$  is a super-item of  $I_2$ .

**Definition 4.** (child and parent items) An item  $I_2$  is a child item of an item  $I_1$  if:

- $I_2$  is a sub-item of  $I_1$
- there is no sub-item  $I$  of  $I_1$  for which  $I_2$  is a sub-item.

If  $I_2$  is a child item of  $I_1$  then  $I_1$  is a parent item of  $I_2$ .

The concept of partial behavioral abstraction can then be introduced.

**Definition 5.** (partial behavioral abstraction) Let  $I$  be an item and  $\mathbb{I} = \{I_1, \dots, I_n\}$  be a set of items.  $I$  is a partial behavioral abstraction of  $\mathbb{I}$  if it exists at least one mode  $m_i \in Modes(I)$ , and a m-proposition  $\mathcal{P}_i$  such as:  $m_i \rightarrow \mathcal{P}_i$  with  $Modes(\mathcal{P}_i) = \{mode(I_1), \dots, mode(I_n)\}$ .

Normally, if a parent-item behaves correctly, it is deduced that its sub-items are in a normal mode. It is represented by a logical implication  $ok(I) \rightarrow ok(I_1) \wedge ok(I_2) \wedge \dots \wedge ok(I_n)$ . In the context of human machine cooperation, partial behavioral abstraction represents the knowledge of expert, who tests the faulty system, about the structure of a system.

**Definition 6.** (complete behavioral abstraction) Let  $I$  be an item and  $\mathbb{I} = \{I_1, \dots, I_n\}$  a set of items.  $I$  is a complete behavioral abstraction of  $\mathbb{I}$  if  $\forall m_i \in Modes(I)$ , it exists a m-proposition  $\mathcal{P}_i$  such as:  $m_i \leftrightarrow \mathcal{P}_i(I_i)$  with  $I_i \in \mathbb{I}$  and  $\bigcup_i I_i = \mathbb{I}$ .

A partial behavioral abstraction  $\mathbb{I} = \{I_1, \dots, I_n\}$  of  $I$  can always be transformed into a complete one in introducing a new virtual item that represents the part of item  $I$  which is not in  $\mathbb{I}$ , denoted by  $VI$  for *virtual item*, with  $VI = \mathbb{I} \setminus \{I\}$ .

In a behavioral abstraction, if a *cfm* mode appears in the support of the m-proposition, the corresponding modes in the super-items are necessary *cfm* because a complementary fault mode cannot lead to a identified fault mode.

### 3.2 Formalizing symptoms

Diagnosis analyzes of physical systems rely on detection tests also called analytical redundancy relations (ARR), that provide symptoms. A detection test consists in checking the consistency between the data collected from a system and the ARRs. A test  $T$  corresponds to a set of modes called support of  $T$  and denoted  $Modes(T)$ . Let  $val(t, V) \in dom(t, V)$  be a set of data-flows (coming usually from measurements

or set points) corresponding to a set of variables  $V$  (see [14], for example). Generally speaking, if  $T$  is an ARR, there exists a m-proposition  $\mathcal{P}$  such as:  $\mathcal{P}$  induces that  $val(t, V)$  is consistent with the ARR  $T$ , with  $Modes(T) = Modes(\mathcal{P})$ . This result is well known in consistency based diagnosis (see [6, 15, 17]). Moreover, a test is *fully checked* if all the data-flows corresponding to  $dom(t, V)$  have been checked. In this case,  $\mathcal{P}$  is equivalent to:  $\forall val(t, V) \in dom(t, V)$ ,  $val(t, V)$  is consistent with the ARR  $T$ . This situation may occur for ARR dealing with variables having discrete value domains: all the possible  $val(t, V) \in dom(t, V)$  can be checked.

Without redundant items, the m-proposition  $\mathcal{P}$  is a conjunction of modes. For example, consider the 3 following items with their sets of behavioral modes:  $Modes(item1) = \{ok, cfm\}$ ,  $Modes(item2) = \{ok, fault1, cfm\}$ ,  $Modes(item3) = \{ok, fault2, cfm\}$ . A inconsistent test may lead to the conclusion:  $\neg(ok(item1) \wedge ok(item2) \wedge fault2(item3))$ .

More details about the computation of tests can be found in [14].

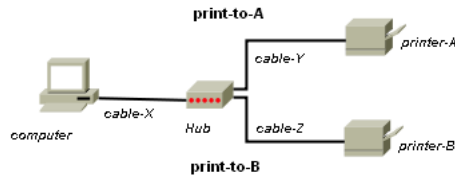
### 3.3 Shared items

In comparison with the abstraction procedure presented in [10, 1, 3], the possibility of shared items is taken into account because decomposition do not necessary leads to tree. A shared item is a child-item that have several parent-items. Taking into account the knowledge coming from the behavioral modes of all the super-items improves the diagnoses in providing better conclusions about the behavioral mode of the shared sub-item. Let's consider the example of a simple system consisting of two printers A et B which are connected to the same computer via a Hub (see figure 2). This *print* function is decomposed into two sub-functions: items *print-to-A* and *print-to-B*. The two items *order-print-to-A* and *order-print-to-B* corresponds to printing orders sent, for instance, by an expert. Consider that an expert performed two tests: firstly, he tries to print to the printer A and did not see any paper coming out from the printer A. The expert infers:

$$\neg ok(print - to - A) \text{ with} \quad (2)$$

$$ok(print - to - A) \rightarrow ok(computer) \wedge ok(cable - X) \wedge ok(Hub) \quad (3)$$

$$\dots \wedge ok(cableY) \wedge ok(printer - A) \wedge ok(order - print - to - A)$$



**Fig. 2** Schema of the printing-system

Then, he succeed in printing to the printer B. He infers:

$$\begin{aligned} ok(\text{print} - to - B) \rightarrow ok(\text{computer}) \wedge ok(\text{cable} - X) \wedge ok(\text{Hub}) \\ \wedge ok(\text{cable} - Z) \wedge ok(\text{printer} - B) \wedge ok(\text{order} - \text{print} - to - B) \end{aligned} \quad (4)$$

But  $ok(\text{print} - to - B)$  cannot be proved (except if the sub-system is considered as fully checked). In this example, it appears clearly that shared items may appear. In this example, it points out that the fault comes up rather from  $\text{cable} - Y$  or from  $\text{printer} - A$

### 3.4 Fault propagation

In actual physical systems, a fault propagation models the fact that a fault (or failure) mode of an item induces fault modes of other items. Fault propagation is usually represented by a logical implication, e.g.  $mode(item_i) \rightarrow mode'(item_j)$ . To take into account fault propagations, the transformation of logical implications into logical conjunctions is preferred. A logical implication  $A \rightarrow B$  is equivalent to  $\neg A \vee B$ , then  $mode(item_i) \rightarrow mode'(item_j)$  is equivalent to  $\neg mode(item_i) \vee mode'(item_j)$ .

## 4 An iterative diagnosis solving process

An iterative diagnosis solving process is proposed: it relies on some well-known fundamental results.

### 4.1 Consistency based approaches *Les parties 4.1 et 4.2 doivent être fusionnées et réduites à maximum 3/4 de page. Le résultat devra être introduit au début de 4.5 Solving algorithm*

[6, 15] have introduced a diagnosis engine based on first-order logic. A system to be diagnosed is defined by (SD, COMPS, OBS), where SD is the system description, COMPS is a set of components, and OBS is a set of observations. A diagnosis is a set of faulty components represented simply by  $AB(C)$ . Then, in [7], this approach has been extended to multiple behavioral modes. Knowledge about failure modes is managed. A computational technique which focuses on prior probability is provided. Thus, a diagnosis is represented by a set that assigns behavioral modes to each component. Component description with multiple behavioral modes associating to constraints has been extended in [4] too, in which,  $SD = CD \cup BD \cup \Gamma$ . CD (compositional description) represents the structure

of the system.  $\Gamma$  represents the general knowledge. BD (behavioral description) represent the behavioral of components in the system, e.g let's take example of a pipe:  $modes(pipe1) = \{ok, stuck, leak, cfm\}$ . Each behavioral mode is modeled by one or more constraints represented by a first-order formula, e.g.  $pipe(pipe1, [in, out]) \supset [leak(pipe1) \supset (in > 0) \wedge (in > out)]$  (see in detail in [4]).

These approaches are presented in order to introduce new concepts that will allow the extension to decomposition during iterative diagnosis process.

**Conflicts/Tests:** A *conflict* is a conjunctive proposition of component behavioral modes which is inconsistent with the system description SD and the corresponding observations. A *conflict set* is the union of the complementary modes of modes appearing in a test. It is minimal if no proper subset of it is a conflict set. In hybrid approach [5], a conflict corresponds to an inconsistent test. Let's review the example of pipe1. Assume that there are observations  $in = out$ , then the test:  $leak(pipe1)$  leads to a conflict. It can be written  $\neg(leak(pipe1)) \leftrightarrow ok(pipe1) \vee stuck(pipe1) \vee cfm(pipe1)$ . Then, a conflict set is obtained  $\{ok(pipe1), stuck(pipe1), cfm(pipe1)\}$ .

**Hitting sets:** Let  $C$  be a collection of conflict sets. A hitting set for  $C$  is a set  $H \subseteq \bigcup_{S \in C} S$  such that  $H \cap S \neq \{\emptyset\}$  for each  $S \in C$ . A hitting set for  $C$  is minimal if no proper subset of it is a hitting set for  $C$ .

## 4.2 Extension to multi-level diagnosis

This section focuses on taking into account models of abstraction. An MHS-Tree (Modify Hitting Set-Tree) algorithm is proposed.

### 4.2.1 Formulation of a complete diagnosis problem

Let's now summarize firstly results that can appears in the statement of a complete diagnosis problem

1. the list of possible modes for items.
2. the behavioral abstractions. Initially, partial behavioural abstraction are modelled by a set of implicative propositions. Then it is transformed into complete behavioural abstraction, and is represented by a set of equivalent propositions. This step will be introduced in detail afterward.
3. the modes implied in fully checked consistent tests, modelled by conjunctive m-propositions.
4. the modes implied in inconsistent tests, modelled by disjunctive m-propositions.
5. the fault propagations, modelled by disjunctive m-propositions.

### 4.2.2 Managing multiple modes

When items contain multiple modes, the standard HS-tree algorithm (a tree whose nodes are hitting sets [15]) may lead to diagnoses that contain several behavioral modes of the same item. However, these diagnoses are impossible because an item may be in only one mode at the same time. To avoid this situation, the notion of *path covering* is introduced. Let  $H$  be a hitting set of modes. The path to root of  $H$  is the set of modes marking the edges that links  $H$  to the root conflict set in the HS-tree, denoted by  $PC(H(n))$ . Consider, for instance, the path  $\{mode_1(I_1), mode_2(I_2), \dots, mode_n(I_n)\}$  the path covering of  $H$ , denoted  $covering(H)$ , is defined by:  $Modes(I_1) \cap \dots \cap Modes(I_n)$ . Hence, multiple modes can be managed in the following way. In standard approaches, when developing a HS-tree, each hitting set  $H$  yields successor branches labelled by the modes of  $Modes(H)$ . To avoid the coexistence of several modes in diagnoses, when developing a node  $H$ , only the modes of  $Modes(H)$  that do not belong to  $covering(H)$  yields new branches.

### 4.2.3 Managing fully checked tests

Whereas standard tests lead to conflict sets only in case of inconsistency, fully checked tests lead also to m-propositions in case of consistency. Assuming the absence of material redundancy. These m-propositions are conjunctions of modes if there is no redundant items. In this case, all the modes within the m-propositions are considered as actual.

## 4.3 Diagnosis process *Enlever la subsection et mettre en introduction de 4*

Let's now detail the process of diagnosis method based on iterative decompositions (top-down method). It is an interactive process between a diagnosis tool (a machine) and an expert. The diagnosis process begins when a malfunction is detected. Fault localisation usually starts with the global tests which tend to test the global function of the system. In each expert's interaction, based on available data and tests, expert does some tests, collects new data and continues the process. According to the monotony principle, the diagnosis tool provides more and more detailed diagnoses as much as new results arise. Step by step, it locates the subsystems or components which are in faulty mode. This diagnosis process is depicted by figure 3.

Note that, between each iteration, the solving process is the same. Let's focus now on what happens between two iterations. Diagnosis process between two iterations can be decomposed into two parts. The first one is called *transformation*, it transforms the expert problem with partial behavioral abstractions into a solvable

problem. The second one is based on a *MHS-Tree algorithm* which computes and provides diagnoses from the solvable problem.

#### 4.4 Transformation

During the transformation step, the initial knowledge about system (symptoms, decomposition model and fault propagations) can be transformed into a m-proposition by:

1. introducing complementary fault mode for each known item
2. introducing virtual complementary items in order to transform partial behavioral abstractions into complete behavioral abstractions in formalizing all the implications from conjunction of child modes to each parent node, in order to compute the corresponding equivalent m-propositions.
3. transforming logical implications from fault propagation into disjunctive propositions (see 3.4).
4. replacing the abstract modes by their equivalent m-propositions for points (3) to (5) in section ??.
5. developing the m-propositions into a conjunctive normal form and splitting the resulting proposition into a set of monomials.

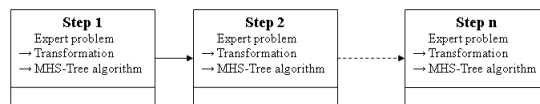
**A développer : trop court. Il faut formaliser proprement (une page et demi)**

Finally, after these transformations, the diagnosis problem to be solved may be formulated as m-proposition whose monomials are provided to the solving algorithm to compute diagnoses.

#### 4.5 Solving algorithm

Let  $\mathcal{P}$  be a m-proposition summarizing a transformed multi abstraction level diagnosis problem. Let  $C$  be the set of all monomials of  $\mathcal{P}$ . Let  $S$  be an element of  $C$

A solving algorithm based on MHS-Tree is presented in this section. It aims at computing diagnoses keeping MHS-Tree as small as possible.



**Fig. 3** Diagnosis process

### 4.5.1 MHS-Tree algorithm

The method consists in generating all modes of a MHS-tree according to a breadth first strategy such as in [15]:

1. *Compute node labels:* If  $\exists S \in C$  such that  $H(n) \cap S = \emptyset$ , the node  $n$  is labelled by  $S$ , if not,  $n$  is labelled by  $\checkmark$ .
2. *Compute branch labels:* If a monomial  $S$  is labelled as a node  $n$ , then successor branch are labelled by elements of  $S$ . A  $mode(I) \in S$  is satisfying for labelling if  $mode(I) \notin PC(H(n))$ , if not, successor branch is pruned.
3. *Tree pruning* If node  $n$  has been generated and there is an other node  $n'$  such that  $H(n) = H(n')$  then close  $n'$ . A closed node is labelled by  $\times$ . If a node  $n$  is labelled by  $\checkmark$ , and there is an other node  $n''$  such that  $H(n) \subseteq H(n'')$ , then close the node  $n''$  by  $\times$ . If a node  $n$  is labelled by  $\checkmark$ , and there is another node  $n''$  which is labelled by  $\checkmark$  such that  $H(n'') \subset H(n)$ , then node  $n$  is closed. In the other word,  $n$  is labelled again by  $\times$ .

In addition to standard HS-tree approaches, the multi-mode context has to be taken into account. It is not a new problem. An solving approach has for instance been proposed in [16]. Alternatively, HS-Tree based algorithm ([15]) is preferred here to manage multiple-modes. **Explain why it is preferred** Moreover, in comparison with original HS-Tree algorithm, which base on a set of conflicts, MHS-Tree is extended to a set of disjunctive propositions to computes hitting set. Each disjunctive proposition can correspond to a test inconsistent or to transformed fault propagation.

In order to keep a sound reasoning, a consistent test is not taken into account to compute diagnoses except if it is fully checked. However, results of normal consistent tests are useful for classification of diagnoses. In [13], an approach based on a distance between theoretical and effective signatures has been proposed. Here, it is extended to multi-mode context.

*Signature of fault mode:* Let  $T = (t_i)$  be an ordered list of tests, and  $M = (m_i)$  be a set of faulty modes. the signature of  $M$  in  $T$  is givent by  $\sigma_T(M)$ :

$$\forall i, \begin{cases} (\sigma_T(M))_i = 1 \leftrightarrow M \cap \overline{\prod_{mode}(t_i)} \neq \emptyset \\ (\sigma_T(M))_i = 0 \leftrightarrow M \cap \prod_{mode}(t_i) = \emptyset \end{cases}$$

where  $\prod_{mode}(t_i)$  corresponds to the set of modes implied in the test  $t_i$ . And  $\overline{\prod_{mode}(t_i)}$  corresponds to the union of complementary modes of each mode implied in the test  $t_i$ :

$$\overline{\prod_{mode}(t_i)} = \bigcup_{m(I) \in \prod_{mode}(t_i)} Modes(I) \setminus \{m(I)\}$$

*Effective signature* Let  $T = (t_i)$  be an ordered list of tests. At an given instant, the effective signature in  $T$ , denoted by  $\sigma_T^*$ , is given by:

$$\forall i, \begin{cases} (\sigma_T^*)_i = 1 \leftrightarrow t_i \text{ is inconsistent} \\ (\sigma_T^*)_i = 0 \leftrightarrow t_i \text{ is consistent} \end{cases}$$

*Plausibility measurement:* The plausibility measurement attempts to measure the similarity between the effective signature and the theoretical signature of a diagnosis [13]. Let  $T = (t_i)$  be an ordered list of tests, and  $D = d_i$  be a set of diagnoses. The plausibility measurement is given by:

$$\forall d_i \in D, \mu_T^c(d_i) = \frac{|\sigma_T(d_i), \sigma_T^*|_{Hamming}}{\dim(T)} \quad (5)$$

Application of the plausibility measurement is illustrated in the next example.

## 5 Application example

In order to illustrate how the proposed approach fits to iterative diagnosis with consecutive decompositions, let's consider a faulty car studied by a car mechanic. Firstly, the car mechanic notes that the car does not start up. At this step, the resulting symptom, which is also a trivial diagnosis, is:  $cfm(car)$ . It is very general and does not direct to the next step: almost every failure is possible. Implicitly, the possible modes for the car are:

$$Modes(car) = \{ok, cfm\} \quad (6)$$

Secondly, because the starting system may be easily checked, the expert implicitly decomposes the car into the electric power resource ( $EPR$ ), the electrical starting system except the starting drive ( $ESS$ ), and the starting drive ( $SD$ ).

The decomposition can be modeled by:

$$ok(car) \rightarrow ok(EPR) \quad (7)$$

$$ok(car) \rightarrow ok(ESS) \quad (8)$$

$$ok(car) \rightarrow ok(SD) \quad (9)$$

Then, the expert turns on the key to test whether the starting drive is operating: it corresponds to a new test. Since he hears the starting drive cranking, he infers from *test 1* that:

$$\exists OBS/ok(EPR) \wedge ok(ESS) \wedge ok(SD) \quad (10)$$

It cannot be generalized to any  $OBS$  in considering this test as fully checked test. Nevertheless, this consistency test can be used to sort the diagnoses using the plausibility measurement. The observed symptoms are now:

$$cfm(car) \quad (11)$$

$$\exists OBS/ok(EPR) \wedge ok(ESS) \wedge ok(SD) \quad (12)$$

The expression (12) means that it exists at least an observation such that the test given by (10) is consistent.

The problem is fully defined by (6), (7), (8), (9), (10), (11) and (12). Let's transform this problem into a solvable problem. In order to obtain a complete behavioral abstraction, complementary fault modes and a virtual item are introduced. It is named:  $VI_1 = car \setminus \{EPR, ESS, SD\}$ . The new transformed set of modes coming from (6) is:

$$Modes(EPR) = (ok, cfm) \quad (13)$$

$$Modes(ESS) = (ok, cfm) \quad (14)$$

$$Modes(SD) = (ok, cfm) \quad (15)$$

$$Modes(VI_1) = (ok, cfm) \quad (16)$$

Decomposition can then be written with equivalences:

$$ok(car) \leftrightarrow ok(EPR) \wedge ok(ESS) \wedge ok(SD) \wedge ok(VI_1) \quad (17)$$

$$cfm(car) \leftrightarrow cfm(EPR) \vee cfm(ESS) \vee cfm(SD) \vee cfm(VI_1) \quad (18)$$

Using the MHS-tree algorithm, the diagnosis of the transformed problem can be computed. It leads to:

$$\{cfm(EPR)\}; \{cfm(ESS)\}; \{cfm(SD)\}; \{cfm(VI_1)\} \quad (19)$$

Diagnoses can now be sorted. A signature table (1) can be obtained from (10), (11) and (12):

**Table 1** Signature table 1

	$ok(EPR)$	$ok(ESS)$	$ok(SD)$	$ok(VI_1)$
$T_1$	1	1	1	0

The theoretical fault signature is:

$$\sigma_T(cfm(EPR)) = (1), \sigma_T(cfm(ESS)) = (1), \sigma_T(cfm(SD)) = (1), \sigma_T(cfm(VI_1)) = (0) \quad (0)$$

Since the test 1 is consistent, the effective signature is  $\sigma_T^* = (0)$ .

From (5), the plausibility measurement is given by:

$$\mu_T^c(cfm(EPR)) = 1.00, \mu_T^c(cfm(ESS)) = 1.00, \mu_T^c(cfm(SD)) = 1.00, \mu_T^c(cfm(VI_1)) = 0.00$$

Because  $\mu_T^c(cfm(VI_1)) = 0.00$  is the lowest value, the expert decides to test sub-parts of the virtual item i.e. parts of the car that are not EPR, ESS or SD. He focuses on the ignition system. The expert disconnects the spark plug with its wires from the car engine, holds the end of spark plug with its wire close to a metal surface and gets help to start up the car without using the starting system. Expert does not see any spark coming from spark plugs. These tests are inconsistent. He infers that the

electric power resource ( $EPR$ ), the ignition circuit ( $IC$ ) or the spark plugs ( $SP$ ) are faulty. The virtual item has thus been decomposed into the ignition circuit ( $IC$ ) and the spark plugs ( $SP$ ):

$$ok(VI_1) \rightarrow ok(SP) \quad (20)$$

$$ok(VI_1) \rightarrow ok(IC) \quad (21)$$

The new test leads to:

$$\neg ok(EPR) \vee \neg ok(SP) \vee \neg ok(IC) \quad (22)$$

Consequently, the new set of symptoms is given by (11), (12) and

$$cfm(VI_1) \quad (23)$$

$$\neg ok(EPR) \vee \neg ok(SP) \vee \neg ok(IC) \quad (24)$$

The new problem to be solved is given by: (6), (7), (8), (9), (10), (11), (12), (20), (21), (22), (23), and (24). The problem is transformed by adding an virtual item  $VI_2 = VI_1 \setminus \{SP, IC\}$ , which is equal to:  $car \setminus \{EPR, ESS, SD, SP, IC\}$ .

The new transformed set of modes is given by (6), (13), (14), (15), (16) and:

$$(SP) = (ok, cfm) \quad (25)$$

$$(IC) = (ok, cfm) \quad (26)$$

$$(VI_2) = (ok, cfm) \quad (27)$$

The transformed abstractions are given by (17), (18) and

$$ok(VI_1) \leftrightarrow ok(SP) \wedge ok(IC) \wedge ok(VI_2) \quad (28)$$

$$cfm(VI_1) \leftrightarrow cfm(SP) \vee cfm(IC) \vee cfm(VI_2) \quad (29)$$

Using the MHS-tree algorithm, the diagnosis of the transformed problem can be computed:

$$\{cfm(EPR)\}; \{cfm(SP)\}; \{cfm(IC)\} \quad (30)$$

From (10), (11) and (12), a signature table is obtained:

**Table 2** Signature table 2

	$ok(EPR)$	$ok(ESS)$	$ok(SD)$	$ok(SP)$	$ok(IC)$	$ok(VI_2)$
$T_1$	1	1	1	0	0	0
$T_2$	1	0	0	1	1	0

The theoretical fault signatures of diagnoses are given by:  $\sigma_T(cfm(EPR)) = (1 \ 1)$ ;  $\sigma_T(cfm(ESS)) = (1 \ 0)$ ;  $\sigma_T(cfm(SD)) = (1 \ 0)$ ;  $\sigma_T(cfm(SP)) = (0 \ 1)$ ;  $\sigma_T(cfm(IC)) = (0 \ 1)$ ;  $\sigma_T(cfm(VI_2)) = (0,0)$  and the effective signature is:  $\sigma_T^* = (0 \ 1)$

Then, the plausibility measurement is given by:

$$\mu_T^c(cfm(EPR)) = 0.50; \mu_T^c(cfm(ESS)) = 1.00; \mu_T^c(cfm(SD)) = 1.00; \mu_T^c(cfm(SP)) = 0.00; \mu_T^c(cfm(IC)) = 0.00; \mu_T^c(cfm(VI_2)) = 0.50.$$

Since  $\mu_T^c(cfm(SP)) = 0.00$ ,  $\mu_T^c(cfm(IC)) = 0.00$  are lowest values, and the ignition circuit seems to be easier to test, expert decides to test subparts of the ignition circuit. The expert's knowledge about the ignition system can be represented in figure 4. The battery (BAT), the ignition switch (ISW) and the ballast resistor (BR) can form a test to check whether power is supplied to the ignition coil. Therefore, expert gets help to start up the car and uses a voltmeter to measure the tension at the point A (see figure 4). The result is  $V_A = 0volt$ . Expert infers that the ignition switch (ISW) or the battery (BAT) or the ballast resistor (BR). The corresponding symptom is:

$$\neg ok(BAT) \vee \neg ok(ISW) \vee \neg ok(BR) \tag{31}$$

The decompositions can be modeled by:

$$ok(IC) \rightarrow ok(ISW) \tag{32}$$

$$ok(IC) \rightarrow ok(BAT) \tag{33}$$

$$ok(IC) \rightarrow ok(BR) \tag{34}$$

$$ok(EPR) \rightarrow ok(BAT)ok(ESS) \rightarrow ok(ISW) \tag{35}$$

Therefore, the symptoms are now represented by (11), (12), (23), (24), and:

$$cfm(IC) \tag{36}$$

$$\neg ok(BAT) \vee \neg ok(ISW) \vee \neg ok(BR) \tag{37}$$

The new problem in this step is defined by (6), (7), (8), (9), (10), (11), (12), (20), (21), (23), (24), (32), (33), (34), (35), (36) and (37). The problem can now be transformed by adding an new item virtual:  $VI_3 = IC \setminus \{ISW, BAT, BR\}$ .

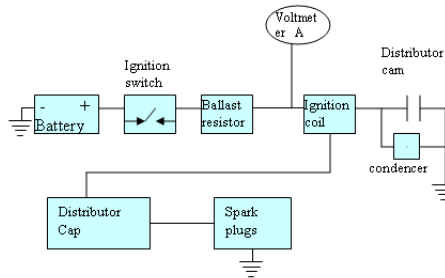


Fig. 4 Diagnosis process

The new transformed set of modes is given by (6), (13), (14), (15), (16), (25), (26), (27) and

$$modes(BAT) = (ok, cfm) \quad (38)$$

$$modes(ISW) = (ok, cfm) \quad (39)$$

$$modes(BR) = (ok, cfm) \quad (40)$$

$$modes(VI_3) = (ok, cfm) \quad (41)$$

The transformed problem is given by (12), (13), (23), (24) and:

$$ok(IS) \leftrightarrow ok(ISW) \wedge ok(BAT) \wedge ok(BR) \wedge ok(VI_3) \quad (42)$$

$$cfm(IS) \leftrightarrow cfm(ISW) \vee cfm(BAT) \vee cfm(BR) \vee cfm(VI_3) \quad (43)$$

Using MHS-tree algorithm, the diagnosis of the transformed problem can be computed:

$$\{cfm(BAT)\}; \{cfm(ISW)\}; \{cfm(BR)\} \quad (44)$$

In order to exploit information from inconsistent test (12), expressions (12), (35) and (35) lead to:

$$\exists OBS/ok(BAT) \wedge ok(EPR \setminus \{BAT\}) \wedge ok(ISW)ok(ESS \setminus \{ISW\}) \wedge ok(SD) \quad (45)$$

From (10), (11), (12), (23), (24), (35), (31), (36), (45), a signature table is obtained:

**Table 3** Signature table 3

	ok(BAT)	ok(EPR ∖BAT)	ok(ISW)	ok(ESS ∖ISW)	ok(SD)	ok(SP)	ok(BR)	ok(IC ∖BAT ∖ISW ∖BR)	ok(VI <sub>3</sub> )
$T_1$	1	1	1	1	1	0	0	0	0
$T_2$	1	1	0	0	0	1	1	1	0
$T_3$	1	0	1	0	0	0	1	0	0

The theoretical signature of diagnoses are:  $\sigma_T(cfm(BAT)) = (1 \ 1 \ 1)$ ;  $\sigma_T(cfm(EPR \setminus \{BAT\})) = (1 \ 1 \ 0)$ ;  $\sigma_T(cfm(ISW)) = (1 \ 0 \ 1)$ ;  $\sigma_T(cfm(ESS \setminus \{ISW\})) = (1 \ 0 \ 0)$ ;  $\sigma_T(cfm(SD)) = (1 \ 0 \ 0)$ ;  $\sigma_T(cfm(SP)) = (0 \ 1 \ 0)$ ;  $\sigma_T(cfm(BR)) = (0 \ 1 \ 1)$ ;  $\sigma_T(cfm(IC \setminus \{BAT, ISW, BR\})) = (0 \ 1 \ 0)$ ;  $\sigma_T(cfm(VI_3)) = (0 \ 0 \ 0)$  and the effective signature:  $\sigma_T^* = (011)$ .

Then, the plausibility measurement is given by:  $\mu_T^c(cfm(BAT)) = 0.33$ ;  $\mu_T^c(cfm(EPR \setminus \{BAT\})) = 0.66$ ;  $\mu_T^c(cfm(ISW)) = 0.66$ ;  $\mu_T^c(cfm(ESS \setminus \{ISW\})) = 1.00$ ;  $\mu_T^c(cfm(SD)) = 1.00$ ;  $\mu_T^c(cfm(SP)) = 0.33$ ;  $\mu_T^c(cfm(BR)) = 0.00$ ;  $\mu_T^c(cfm(IC \setminus \{BAT, ISW, BR\})) = 0.33$ ;  $\mu_T^c(cfm(VI_3)) = 0.66$ .

Since  $\mu_T^c(cfm(BR)) = 0.00$  is the lowest value, expert decides to test the balast resistor with a set of known modes  $modes(BR) = \{ok, blown - out, cfm\}$ , a test is formed. Expert uses a ohmmeter to test the ballast resistor. The result points out that the value is consistent with its reference. It can be considered as a fully checked test. Hence the fault is easily localized:

$$blown - out(BR) \quad (46)$$

## 6 Conclusion

This presented results make it possible to develop human-machine cooperative diagnosis process. It becomes possible to tackle diagnosis problems without having an initial complete model of the system. A top-down iterative process has been proposed to exploit information step by step thank to hierarchical decomposition. Diagnoses are refined step by step. For this purpose, diagnosis problems inferred from the expert knowledge provided at each iteration, are solved by a transformation into a solvable problem composed of the whole available knowledge (decomposition, inconsistent tests, consistent tests, fully checked tests and fault propagation) coming from system modeling. The resulting diagnosis problem can then be solved according to the proposed MHS-tree algorithm. The iterative diagnosis process is illustrated by an example.

## References

- [1] K. Autio. Abstraction of behaviour and structure in model-based diagnosis. In *the Sixth International Workshop on Principle of Diagnosis (DX95)*, pages 1–7, 1995.
- [2] K. Autio and R. Reiter. Structural abstraction in model based diagnosis. In *The 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 269–273. John Wiley and Sons, 1998.
- [3] L. Chittaro and A.N. Kumar. Reasoning about function and its applications to engineering. *Artificial Intelligence in Engineering*, 12(4):331–336, 1998.
- [4] L. Chittaro and R. Ranon. Hierarchical model-based diagnosis based on structural abstraction. *Artificial Intelligence*, 1-2:147–182, 2004.
- [5] M. O. Cordier, P. Dague, M. Dumas, F. Lévy, J. Montmain, M. Staroswiecki, and L. Travé-massuyès. A comparative analysis of ai and control theory approaches to model-based diagnosis. In *14th European Conference on Artificial Intelligence*, Berlin, Germany, 2000.
- [6] J. De Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

- [7] Johan de Kleer and Brian C. Williams. Diagnosis with behavioral modes. *Readings in model-based diagnosis*, pages 124–130, 1992.
- [8] M. Desindes. *Contribution à la mise au point d'une approche intégrée analyse diagnostique / analyse de risques*. PhD thesis, Grenoble Institute of Technology (INPG), 2006.
- [9] R. Greiner, B. A. Smith, and R. W. Wilkerson. A correction to the algorithm in reiter's theory of diagnosis. *Artificial Intelligence*, 41(1):79–88, 1989.
- [10] I. Mozetič. *Hierarchical model-based diagnosis*, pages 354–372. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.
- [11] P. P. Nayak and A. Y. Levy. A semantic theory of abstractions. In *14th International Joint Conference on Artificial Intelligence IJCAI-95*, pages 196–203, Montreal, Canada, 1995.
- [12] S. Ploix and P. Chazot. Iterative expert driven fault diagnosis based on structural modeling. In *IFAC Symposium SAFEPROCESS'2006*, August 2006.
- [13] S. Ploix, S. Touaf, and J. M. Flaus. A logical framework for isolation in fault diagnosis. In *SAFEPROCESS'2003*, Washington D.C., U.S.A., 2003.
- [14] S. Ploix, A. Yassine, and J.-M. Flaus. An improved algorithm for the design of testable subsystems. In *The 17th IFAC World Congress*, Seoul, Corea, 2008.
- [15] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
- [16] Ron Rymon. Search through systematic set enumeration. In *KR*, pages 539–550, 1992.
- [17] P. Struss. What's in sd? towards a theory of modeling for diagnosis. In W. Hamscher, L. Console, and J. De Kleer, editors, *Readings in model-based diagnosis*, pages 419–448. Morgan Kaufman, 1992.