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 levels and partly modelled components would be relevant to actual needs. This paper proposes a diagnosis tool managing different modelling abstraction levels and partly depicted systems.

1 Introduction

In the diagnosis community, abstraction has been presented as a promising technique to reduce the computational cost of model-based diagnosis [6, 1, 2, 3]. Abstract procedure tends to aggregate items to describe a system at different levels of abstraction with different levels of details (structural and behavioral). It is called *bottom-up method* because it begins by the most detailed 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.

In this paper, the term *item* is preferred to *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.

Quang-Huy GIAP · Stephane PLOIX · Jean-Marie FLAUS
Laboratoire G-SCOP, 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 - France,
e-mail: {quang-huy.giap,stephane.ploix,jean-marie.flaus}@g-scop.inpg.fr

Please use the following format when citing this chapter:

Giap, Q.-H., Ploix, S. and Flaus, J.-M., 2009, in IFIP International Federation for Information Processing, Volume 296; *Artificial Intelligence Applications and Innovations III*; Eds. Iliadis, L., Vlahavas, I., Bramer, M.; (Boston: Springer), pp. 407–415.

2 Problem statement

2.1 Behavioural and functional 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 *behavior* of an item is modelled by constraints characterizing the set of possible values of involved variables. The *behavioral mode* of an item is modelled by one or more constraints. In [5], the model of multiple modes is introduced. Then, each item may have a normal mode *ok* and a set of possible abnormal modes including a complementary unknown fault mode *cfm*. A specific fault mode is denoted by *fm*. Hence, the set of behavioral modes of an item may be written: $Modes(item_i) = \{ok, [fm_1, \dots, fm_n], \, cfm\}$

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.

2.2 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 child-items that may be child-functions, child-components, ... In other words, an expected behavioral mode of an item is achieved by its child-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 $\mathscr{P}(mode_1, \dots, mode_n)$ is a m-proposition, the support $\mathscr{P}$ is defined by $Modes(\mathscr{P}) = \{mode_1, \dots, mode_n\}$.

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: $(mode_4 \vee mode_5 \vee mode_2) \wedge mode_3$.

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, $mode_4 \vee mode_5 \vee mode_2$ and $mode_3$ are the monomials of the m-proposition.

The concept of partial behavioral abstraction can then be introduced.

Definition 3. (partial behavioral abstraction) Let I be an item and $\mathbb{I} = \{I_1, \ldots, I_n\}$ a set of items. I is a partial behavioral abstraction of $\mathbb{I}$ if $\forall m_i \in Modes(I)$, it exists a m-proposition $\mathscr{P}_i$ such as: $m_i \rightarrow \mathscr{P}$ with $Modes(\mathscr{P}_i) = \{mode(I_1), \ldots, mode(I_n)\}$.

If I is a partial behavioral abstraction of $\mathbb{I} = \{I_1, \ldots, I_n\}$, I is named *parent-item* of each I_i and each I_i is a child-item of I. Normally, if a parent-item behaves correctly, it is deduced that its child-items are in a normal mode. It is represented by a logical implication $ok(I) \rightarrow ok(I_1) \wedge ok(I_2) \wedge \ldots \wedge ok(I_n)$. 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 4. (complete behavioral abstraction) Let I be an item and $\mathbb{I} = \{I_1, \ldots, I_n\}$ 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 $\mathscr{P}_i$ such as: $m_i \leftrightarrow \mathscr{P}$ with $Modes(\mathscr{P}_i) = \{mode(I_1), \ldots, mode(I_n)\}$.

A partial behavioral abstraction $\mathbb{I} = \{I_1, \ldots, I_n\}$ 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 $VI = I \setminus \mathbb{I}$.

2.3 *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 achieved. A logical implication $A \rightarrow B$ is equivalent to $\neg A \vee B$, then $mode(item_i) \rightarrow mode'(item_j)$ is equivalent to $\neg mode(item_i) \vee mode(item_j)$.

2.4 *Formulation of a complete diagnostic problem*

Let's summarize results that can appears in the statement of a complete diagnostic problem

1. the list of items and possible modes for each item.
2. the partial behavioral abstractions inferred from expert's knowledge.
3. the modes implied in inconsistent tests, modelled by disjunctive m-propositions.
4. the fault propagations, modelled by disjunctive m-propositions.

3 An iterative diagnosis solving process

Let's now detail the diagnosis process based on interactive 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 isolation usually starts with the tests that check the global function of a system. In each expert's interaction, expert performs tests, collects new data and continues the process. According to the monotony principle, the diagnosis tool provides more and more detailed diagnoses as new results arise. Step by step, it locates the subsystems or components which are in a faulty mode. This diagnosis process is depicted by figure 1.

Note that, the solving process is the same at each interaction. Let's focus now on what happens between two interactions. Diagnosis process between two interactions 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.

3.1 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 mode, in order to compute the corresponding equivalent m-propositions.
3. transforming logical implications from fault propagation into disjunctive propositions (see 2.3).
4. replacing the abstract modes by their equivalent m-propositions for points (3) and (4) in section 2.4.
5. developing the m-propositions into a conjunctive normal form and splitting the resulting proposition into a set of monomials.

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.

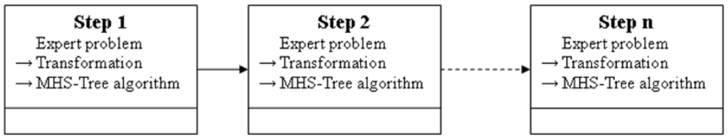

Fig. 1 Diagnosis process

3.2 Solving algorithm

When items contain multiple modes, the standard HS-tree algorithm (a tree whose nodes are hitting sets [8]) 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.

In addition to standard HS-tree approaches, the multi-mode context has to be taken into account. It is not a new problem. In literature, some solving approaches has for instance been proposed in [5, 9]. Based on ATMS [4], the model of faults is integrated in GED+ [9] to analyse whether the faultiness of the components would really explain the observation. In multi-mode context, Sherlock [5] is developed from GDE to compute automatically conflict set and diagnostic hypotheses. It focus reasoning on more probable probabilities firs in attempt to control the combinatorics. Without the constraints propagation technique, HS-Tree based algorithm [8] is preferred in this section to manage multiple-modes. The path from a node to the root node of a HS-Tree show clearly all elements implied in a temporary diagnostic result in the construction of HS-Tree. Then, it is easy to avoid the existence of two or more modes of an Item in a diagnostic result. 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 [7], an approach based on a distance between theoretical and effective signatures has been proposed. Here, it is extended to multi-mode context.

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 given by $\sigma_T(M)$:

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

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:

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

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

$$\forall i, \begin{cases} (\sigma_T^*)_i = 1 \leftrightarrow t_i \text{ is inconsistent} \\ (\sigma_T^*)_i = 0 \leftrightarrow t_i \text{ is consistent} \end{cases} \tag{2}$$

The next measurement attempts to measure the similarity between the effective signature and the theoretical signature of a diagnosis [7]. Let $T = (t_i)$ be an ordered list of tests, and $D = d_i$ be a set of diagnoses. The coincidence measurement is given by:

$$\forall d_i \in D, \mu_T^c(d_i) = \frac{|\sigma_T(d_i), \sigma_T^*|_{Hamming}}{dim(T)} \tag{3}$$

Application of this measurement is illustrated in the next example.

4 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:

$$Modes(car) = \{ok,\ cfm\} \tag{4}$$

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 modelled by:

$$ok(car) \rightarrow ok(EPR) \wedge ok(ESS) \wedge ok(SD) \tag{5}$$

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:

$$\exists OBS/ok(EPR) \wedge ok(ESS) \wedge ok(SD) \tag{6}$$

The consistency test can be used to sort the diagnoses using the coincidence measurement. The observed symptoms are now:

$$cfm(car) \tag{7}$$

$$\exists OBS/ok(EPR) \wedge ok(ESS) \wedge ok(SD) \tag{8}$$

Expression (8) means that it exists at least an observation such that the test given by (6) is consistent.

The problem is fully defined by (4), (5), (6), (7) and (8). 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 (4) is:

$$\begin{aligned}\{Modes(EPR) = (ok, cfm);\ Modes(ESS) = (ok, cfm);\\ Modes(SD) = (ok, cfm);\ Modes(VI_1) = (ok, cfm)\}\end{aligned} \tag{9}$$

Decomposition can then be written with equivalences:

$$ok(car) \leftrightarrow ok(EPR) \wedge ok(ESS) \wedge ok(SD) \wedge ok(VI_1) \tag{10}$$

$$cfm(car) \leftrightarrow cfm(EPR) \vee cfm(ESS) \vee cfm(SD) \vee cfm(VI_1) \tag{11}$$

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

$$\{cfm(EPR)\}; \{cfm(ESS)\}; \{cfm(SD)\}; \{cfm(VI_1)\} \tag{12}$$

Diagnoses can now be sorted. A signature table (1) can be obtained from (6), (7) and (8):

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: $\sigma_T(cfm(EPR)) = (1)$; $\sigma_T(cfm(ECS)) = (1)$; $\sigma_T(cfm(SM)) = (1)$; $\sigma_T(cfm(VI_1)) = (0)$. Since the test 1 is consistent, the effective signature is $\sigma_T^* = (0)$. From (3), the coincidence measurement is given by: $\mu_T^c(cfm(EPR)) = 1.00$; $\mu_T^c(cfm(ECS)) = 1.00$; $\mu_T^c(cfm(SM)) = 1.00$; $\mu_T^c(cfm(VI_1)) = 0.00$. 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*):

$$ok(VI_1) \rightarrow ok(SP) \wedge ok(IC) \tag{13}$$

The new test leads to:

$$\neg ok(EPR) \vee \neg ok(SP) \vee \neg ok(IC) \tag{14}$$

Consequently, the new set of symptoms is given by (7), (8) and

$$cfm(VI_1) \tag{15}$$

$$\neg ok(EPR) \vee \neg ok(SP) \vee \neg ok(IC) \tag{16}$$

The new problem to be solved is given by: (4), (5), (6), (7), (8), (13), (14), (15), and (16). 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 (4), (9)and:

$$\{(SP) = (ok, cfm);\ (IC) = (ok, cfm);\ (VI_2) = (ok, cfm)\} \tag{17}$$

The transformed abstractions are given by (10), (11) and

$$ok(VI_1) \leftrightarrow ok(SP) \wedge ok(IC) \wedge ok(VI_2) \tag{18}$$

$$cfm(VI_1) \leftrightarrow cfm(SP) \vee cfm(IC) \vee cfm(VI_2) \tag{19}$$

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

$$\{cfm(EPR)\};\ \{cfm(SP)\};\ \{cfm(IC)\} \tag{20}$$

From (6), (7) and (8), 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: $\sigma_T(cfm(EPR)) = (1 \quad 1)$; $\sigma_T(cfm(ESS)) = (1 \quad 0)$; $\sigma_T(cfm(SD)) = (1 \quad 0)$; $\sigma_T(cfm(SP)) = (0 \quad 1)$; $\sigma_T(cfm(IC)) = (0 \quad 1)$; $\sigma_T(cfm(VI_2)) = (0,0)$. And the effective signature is: $\sigma_T^* = (0 \quad 1)$.

Then, the coincidence 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, in the end of this step, the faulty part is localized at the ignition circuit (IC) or at the spark plugs (SP).

5 Conclusion

This proposed approach makes it possible to develop human-machine cooperative diagnosis process to tackle diagnosis problems without having an initial complete model of the system. A top-down iterative process has been proposed to handle 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 transformations into a solvable problems composed of the available knowledge (decomposition, inconsistent 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 R. Ranon. Hierarchical model-based diagnosis based on structural abstraction. *Artificial Intelligence*, 1-2:147–182, 2004.

[4] Johan de Kleer. Problem solving with the atms. *Artif. Intell.*, 28(2):197–224, 1986.

[5] Johan de Kleer and Brian C. Williams. Diagnosis with behavioral modes. *Readings in model-based diagnosis*, pages 124–130, 1992.

[6] I. Mozetic. *Hierarchical model-based diagnosis*, pages 354–372. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[7] 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.

[8] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[9] Peter Struss and Oskar Dressler. "physical negation" integrating fault models into the general diagnostic engine. pages 1318–1323, 1989.