Available online at www.sciencedirect.com
SCIENCE DIRECT®
Artificial Intelligence 155 (2004) 147–182
Artificial Intelligence
www.elsevier.com/locate/artint

Hierarchical model-based diagnosis based on structural abstraction

Luca Chittaro *, Roberto Ranon

Department of Mathematics and Computer Science, University of Udine, via delle Scienze 206, 33100 Udine, Italy

Received 9 December 2002; received in revised form 20 June 2003

Abstract

Abstraction has been advocated as one of the main remedies for the computational complexity of model-based diagnosis. However, after the seminal work published in the early nineties, little research has been devoted to this topic. In this paper, we consider one of the types of abstraction commonly used in diagnosis, i.e., structural abstraction, investigating it both from a theoretical and practical point of view. First, we provide a new formalization for structural abstraction that generalizes and extends previous ones. Then, we present two new different techniques for model-based diagnosis that automatically derive easier-to-diagnose versions of a (hierarchical) diagnosis problem on the basis of the available observations. The two proposed techniques are formulated as extensions of the well-known Mozetic's algorithm [I. Mozetic, Hierarchical diagnosis, in: W.H.L. Console, J. de Kleer (Eds.), Readings in Model-Based Diagnosis, Morgan Kaufmann, San Mateo, CA, 1992, pp. 354–372], and experimentally contrasted with it to evaluate the obtained efficiency gains.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Model-based diagnosis; Abstraction; Hierarchical reasoning

1. Introduction

In the last decade, *Model-Based Diagnosis* (*MBD*) [4,6,9,18] has been a very active area of research in Artificial Intelligence that has led also to significant industrial projects (e.g., [16,20,22]).

* Corresponding author.
E-mail addresses: chittaro@dimi.uniud.it (L. Chittaro), ranon@dimi.uniud.it (R. Ranon).

0004-3702/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2003.06.003

Computational complexity of multiple fault diagnosis is one of the well-known problems that needs to be tackled in order to deploy real-world applications of MBD.

Since the late eighties, some researchers (e.g., [10,12,14,19]) advocated *abstraction* as one of the main remedies for this problem. The proposed approaches are typically *hierarchical*: they represent the problem at multiple levels of detail, and then isolate faults one level at a time, starting at the most abstract possible level and using the results at one level to focus reasoning at more detailed levels, thus reducing the overall computational cost of diagnosis.

Two kind of abstractions are commonly employed in MBD: *structural abstraction* [5, 10], which aggregates components to describe the system at different levels of structural detail, and *behavioral abstraction* [12,14,19], which applies simplification operators to describe the system at different levels of behavioral detail (e.g., moving from quantitative to qualitative values in describing the functioning of components).

In this paper, we build on seminal work on abstraction in MBD, and investigate structural abstraction both from a theoretical and practical point of view.

First, we provide a new logical formalization for structural abstraction that generalizes and extends previous ones [1,14,19]. Our proposal builds on the well-known consistency-based theory of diagnosis [8] and on a general framework (the semantic theory of abstraction [15]) for the representation of abstraction between first-order theories. Unlike previous formalizations of structural abstraction, our proposal allows one to represent components with multiple behavioral modes (e.g., valves). Thus, it can be employed with a wider class of physical systems. Moreover, the proposed formalization allows one to easily prove some properties of structural abstraction that are useful in diagnostic reasoning.

Second, we present two new techniques for hierarchical model-based diagnosis that are able to automatically derive easier-to-diagnose versions of a given diagnosis problem on the basis of the available observations. The goal of our research is to move from simply using abstraction in diagnosis to *using a good abstraction for the situation at hand*, i.e., using context to choose it. Indeed, one limit to the effectiveness of current approaches to hierarchical diagnosis is the fact that a single, pre-set hierarchical representation is employed, regardless of the currently available diagnostic information. In some cases, this leads to suboptimal or even counterproductive results in terms of efficiency (some detailed examples will be illustrated in the paper). Unfortunately, most abstractions are usually manually engineered, and thus building a suitable abstract system representation for each diagnostic scenario is not a viable solution. We tackle the problem by using the idea of automatically tailoring an existing multi-level abstraction hierarchy (that may come from design) to the particular problem at hand. The two techniques (called *REARRANGE* and *BOTTOM-UP*) we developed for this purpose:

- are based on different strategies, and can be easily combined together to sum up their respective advantages;
- build on the seminal work on hierarchical diagnosis by Mozetic [14], and are presented as extensions to that approach.

To evaluate the efficiency gains that can be obtained, we present a detailed experimental evaluation using a set of different hydraulic systems, considering the original Mozetic's approach, the two techniques in isolation, and the two techniques in combination.

The techniques presented in the paper are general, and can be easily adopted by any model-based approach that follows the widely adopted consistency-based paradigm. The paper illustrates in detail the algorithms that implement the proposed techniques to allow interested readers to easily include them and experiment with them in their systems.

Finally, this work builds also on previous approaches we proposed in the domain of Flow-Based Functional Models [3,17] and, more generally, in the context of structural abstraction [2]. This paper extends and improves the latter in several directions, in particular:

- it proposes a proper theoretical framework, i.e., the formalization for structural abstraction mentioned above;
- it discusses (using also detailed examples) why using the same hierarchical representation regardless of the currently available diagnostic information can limit or even eliminate the efficiency gains of abstraction;
- it refines the *REARRANGE* technique and proposes a new technique for hierarchical diagnosis (i.e., the *BOTTOM-UP* technique);
- it presents a detailed experimental comparison of the proposed techniques and the reference approach of Mozetic.

The paper is structured as follows: Section 2 summarizes background work on which we build; Section 3 defines a formalization of structural abstraction in diagnosis and illustrates its properties, discussing also related work; Section 4 considers the problem of diagnostic reasoning with structural abstraction by illustrating the state of the art, proposing methods to improve it by automatically tailoring existing abstractions to the situation at hand, and finally illustrating the experimental activity that has been carried out. Section 5 concludes the paper by presenting some possibilities for further work.

2. Background

In this section, we briefly illustrate those aspects of the consistency-based theory of diagnosis [8] and the semantic theory of abstraction [15] that are relevant to illustrate our work, and add some minor extensions to them. Moreover, we clarify the concepts by introducing detailed examples taken from the hydraulic domain that will be followed throughout the paper.

2.1. Representing diagnosis problems

Following [8], a diagnosis problem D in a language L is defined as a triple (*SD*, *OBS*, *COMPS*) where *SD* and *OBS* are first-order theories in language L, representing the system description and observations, respectively, and *COMPS* is a subset of the object constants of L, i.e., the names of the system components.

In order to explicitly separate structural knowledge (i.e., how components are connected together) from behavioral knowledge (i.e., how components behave), we divide *SD* in the following way:

$$SD = CD \cup BD \cup \Gamma$$

where *BD* (*behavioral description*) represents the behavior of the components in the system, *CD* (*compositional description*) represents the structure of the system, and Γ represents general knowledge (e.g., hydraulic laws) that is not specific to the considered system.[1]

Definition 1 (*behavioral description of a component*). The behavioral description of a component c (denoted BD_c) in a diagnosis problem D, is a set of sentences of the form

$$T_c(c, \vec{z}_c) \supset \left[m(c) \supset \sigma_c(\vec{z}_c)\right]$$

where

- $T_c(c, \vec{z}_c)$ is the component type predicate for c, and $\vec{z}_c$ lists the ports of the component,
- m is a predicate identifying one of the behavioral modes of c,
- $\sigma_c(\vec{z}_c)$ is a first-order formula with free variables $\vec{z}_c$ describing the behavioral mode m of c with respect to its ports.

The behavior of a component is thus represented as a set of first-order sentences, each one corresponding either to a normal or a faulty behavior, which is defined by a predicate over the component ports.

Definition 2 (*behavioral description*). The behavioral description *BD* of a diagnosis problem D is the union of the behavioral descriptions of the components in *COMPS*.

Definition 3 (*compositional description*). The compositional description *CD* of a diagnosis problem $D = (SD, OBS, COMPS)$ is a first-order sentence of the form

$$T(S, \vec{z}_S) \equiv \exists z_1, z_2, \ldots, z_n \left(\bigwedge_{c \in COMPS} T_c(c, \vec{z}_c) \right)$$

where

- S is the system's name,
- every variable in $\vec{z}_S$ appears in one of the tuples $\vec{z}_c$, for any $c \in COMPS$, and other variables in the tuples are $z_1, z_2, \ldots, z_n$.

The right-hand part of compositional description *CD* contains one component type predicate for each component in the system. A connection between two components is

[1] A similar separation of structural and behavioral knowledge was proposed in [1]. Here we extend it to components with multiple behavioral modes.

represented by using the same variable name for the two connected ports (one belonging to the first component, the other to the second one). The left hand part of *CD* is composed by a component type predicate for the whole system, which lists those component ports which do not connect two components, but connect a component to the system's environment.

In general, we assume that observations on the system are taken at component's ports.

We adopt the following standard definitions employed in consistency-based diagnosis. Given a diagnosis problem D, a *candidate* C is a formula that assigns to each component of D one of its behavioral modes, while a *partial candidate* is a formula assigning a behavioral mode only to some components of D. A (partial) candidate C is a (*partial*) *diagnosis* if it is logically consistent with both *SD* and *OBS*, i.e., there is an assignment I of values to the ports of the system such that

$$I \models SD \wedge OBS \wedge C$$

In this case, we will call I a model of D; similarly, we will say that D is inconsistent if it has no models. To solve the diagnosis problem, one should find all diagnoses, or a compact characterization for them (such as the *kernel diagnoses* described in [8]).

Finally, we will use the notation $CANDS(D, B)$, where $B \subseteq COMPS$, to denote the set of all (partial) candidates for the diagnosis problem D that can be built using all and only the components in set B. The cardinality of $CANDS(D, COMPS)$ depends both on the cardinality of *COMPS*, and on the number of possible behavioral modes which are defined for the components in *BD*. If a component c_i of a diagnosis problem has M_i possible behavioral modes, then

$$\left|CANDS(D, COMPS)\right| = \prod_{c_i \in COMPS} M_i$$

resulting in a number of candidates which is exponential in $|COMPS|$. Considering the simplest case where each component has only two behavioral modes (ok and faulty), the number of candidates is 2^n, where n is the number of components. More realistic cases (where there are more than two modes) will lead to larger cardinalities.

Example 1. The simple hydraulic system depicted in Fig. 1, called *hydraulic case study* (*hcs*) hereinafter, contains 11 components: a volumetric pump *pm* (delivering a constant flow equal to F_K), pipes p_1, p_2, p_3, p_4, p_5, p_6, valves v_1 and v_2, and three-way nodes n_1 and n_2. The ports in the system are: $t_1, \ldots, t_{13}$ (connecting components together), s_{v_1} and s_{v_2} (allowing one to set the state—open or closed—of valves v_1 and v_2, respectively).

We suppose that each component has one *ok* behavioral mode, which represents its normal behavior. The considered faults are: external leaks (denoted by the predicate *leak*), which affect pumps, pipes, and valves; stuck-at-closed and stuck-at-open faults (denoted by the predicates *stuckC* and *stuckO*, respectively), which affect valves; low or high delivered flowrates (denoted by the predicates *loF* and *hiF*), which affect pumps.

The behavioral descriptions of the components are given in Table 1. For simplicity, we suppose that three-way nodes cannot be faulty, i.e., they have only a normal behavioral mode. Moreover, we consider two component types: one for three-way nodes with one intended input and two outputs, called *twoOut*, and one for three-way nodes with two intended inputs and one output, called *twoIn*.

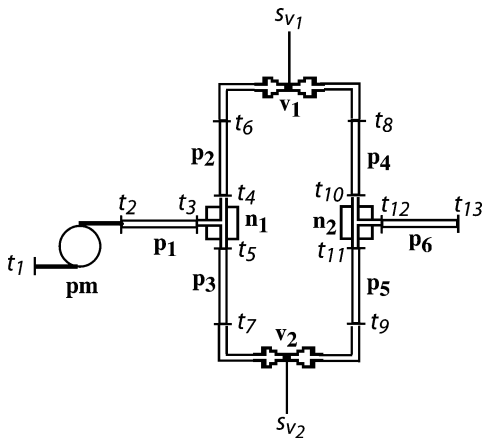

Fig. 1. Layout of the hydraulic case study (component names are shown in bold).

Table 1
Behavioral descriptions for components in the *hcs*

Component	Type predicate	Behavioral modes
pump	$pump(x, [in, out])$	$ok(x) \supset in = out = F_K$ $leak(x) \supset in > out$ $loF(x) \supset in = out < F_K$ $hiF(x) \supset in = out > F_K$
pipe	$pipe(x, [in, out])$	$ok(x) \supset in = out$ $leak(x) \supset in > out$
valve	$valve(x, [s, in, out])$	$ok(x) \supset [(s = closed \wedge in = out = 0) \vee (s = open \wedge in = out)]$ $leak(x) \supset in > out$ $stuckO(x) \supset (s = closed \wedge in = out > 0)$ $stuckC(x) \supset (s = open \wedge in = out = 0)$
three-way node	$twoOut(x, [in, out_1, out_2])$ $twoIn(x, [in_1, in_2, out])$	$ok(x) \supset in = out_1 + out_2$ $ok(x) \supset in_1 + in_2 = out$

The compositional description *CD* for the *hcs* is the following formula:

$$
\begin{aligned}
&hydSystem\big(hcs, [t_1, t_{13}, s_{v_1}, s_{v_2}]\big) \equiv \exists t_2, \ldots, t_{12}\big[pump(pm, [t_1, t_2]) \wedge \\
&\quad pipe(p_1, [t_2, t_3]) \wedge twoOut(n_1, [t_3, t_4, t_5]) \wedge pipe(p_2, [t_4, t_6]) \wedge \\
&\quad pipe(p_3, [t_5, t_7]) \wedge valve(v_1, [s_{v_1}, t_6, t_8]) \wedge valve(v_2, [s_{v_2}, t_7, t_9]) \wedge \\
&\quad pipe(p_4, [t_8, t_{10}]) \wedge pipe(p_5, [t_9, t_{11}]) \wedge twoIn(n_2, [t_{10}, t_{11}, t_{12}]) \wedge \\
&\quad pipe(p_6, [t_{12}, t_{13}])\big]
\end{aligned}
$$

Finally, for this system, $|CANDS(D, COMPS)| = 4 \cdot 2^6 \cdot 4^2 = 4096$.

2.2. Representing abstraction

We adopt the semantic theory of abstraction [15], viewing abstractions as model level mappings. From this perspective, abstracting a theory is a two-step process that first abstracts the intended domain model, and then builds an abstract theory to capture the abstracted domain model. Although there are also other theories that could be used for our purposes (see, e.g., [11]), the theory we adopt has the advantage of explicitly representing the motivation and choices behind a given abstraction.

More formally, given U_0 and U_1, sets of sentences[2] in languages L_0 and L_1, then an *abstraction mapping* π is a function that maps models of a (base) theory U_0 to corresponding interpretations of an (abstract) language L_1:

$$\pi : \mathit{Models}(U_0) \rightarrow \mathit{Interpretations}(L_1)$$

Given any model M_0 of U_0, π builds the interpretation $\pi(M_0)$ for L_1 by defining:

- a formula $\pi_\forall$ (with one free variable, x_1), that defines the abstract universe;
- for each n-ary relation R in L_1, a formula $\pi_R \in L_0$ with n free variables $x_1, \ldots, x_n$, that defines R. Given any model M_0 of U_0, π_R defines an n-ary relation in M_0. The denotation of R in $\pi(M_0)$ is π_R restricted to the universe of $\pi(M_0)$;
- similar formulas to specify the denotation of abstract object and function constants (see [15] for more details).

U_1 is an abstraction of U_0, given the abstraction mapping π, if and only if U_1 captures all the abstract models (with respect to π) of U_0. This is captured by the definition of *model increasing* (MI) abstractions:

Definition 4 (*model increasing abstraction*). Let U_0 and U_1 be sets of sentences in languages L_0 and L_1, respectively. Let $\pi : \mathit{Models}(U_0) \rightarrow \mathit{Interpretations}(L_1)$ be an abstraction mapping. U_1 is a *Model Increasing* (*MI*) *Abstraction of* U_0 with respect to π (written $U_1 <_\pi U_0$) if for every model M_0 of U_0, $\pi(M_0)$ is a model of U_1.

MI abstractions cover various common model level abstractions, such as taking the union of a set of predicates (see [15] for examples of other MI abstractions).

Example 2. Consider the behavioral description for a valve that has been given in Example 1. Suppose that we want to build a more abstract description where we want to represent only two fault modes instead of three: one (*faultyO*), covers all the faults in the open state, and the other (*faultyC*), covers all the faults in the closed state, i.e.:

$$\mathit{valve}\big(x, [s, \mathit{in}, \mathit{out}]\big) \supset \big[\mathit{faultyO}(x) \supset (s = \mathit{open}) \wedge (\mathit{in} > \mathit{out} \vee \mathit{in} = \mathit{out} = 0)\big],$$
$$\mathit{valve}\big(x, [s, \mathit{in}, \mathit{out}]\big) \supset \big[\mathit{faultyC}(x) \supset (s = \mathit{closed}) \wedge (\mathit{in} \neq 0 \vee \mathit{out} \neq 0)\big]$$

[2] We adopt the convention of using a greater subscript to indicate a more abstract, simpler theory.

The abstraction mapping we must adopt is such that the denotation of predicate *faultyO*(x) is the union of the denotations of predicates *leak* (restricted to the cases where $s = open$) and *stuckC*, while the denotation of predicate *faultyC*(x) is the union of the denotations of predicates *leak* (restricted to the cases where $s = closed$) and *stuckO* of Example 1, i.e.,

$$faultyO(x) \equiv_\pi s = open \wedge \big(leak(x) \vee stuckC(x)\big),$$
$$faultyC(x) \equiv_\pi s = closed \wedge \big(leak(x) \vee stuckO(x)\big)$$

Other kind of abstractions, which are not MI, but can be useful in reasoning, can be viewed as MI abstractions in conjunction with *simplifying assumptions* [15], i.e., U_1 is a MI abstraction of U_0 under simplifying assumption A if there is a theory $U_A \subseteq U_0$ such that U_A is consistent with A and U_1 is a MI abstraction of $U_A \cup A$.

We are interested in the following properties of MI abstractions (see [15] for more details):

Proposition 1. *Let U_1 be a MI abstraction of U_0. If U_1 is inconsistent, then U_0 is inconsistent.*

Proposition 1 implies that to prove the inconsistency of a base theory, it suffices to prove the inconsistency of a (potentially simpler) MI abstraction of that theory (in general, the converse does not hold).

Proposition 2. *Let U_1 be a MI abstraction of U_0 and V_1 be a MI abstraction of V_0 under the same interpretation mapping π. Then $U_1 \cup V_1$ is a MI abstraction of $U_0 \cup V_0$ under π.*

Proposition 2 implies that MI abstractions are compositional, allowing one to build abstract theories by composing knowledge from different sources.

3. Formalizing structural abstraction

In this section, we propose a formalization for the concept of structural abstraction in diagnosis. We will define the structural abstraction of a diagnosis problem D as a (more abstract) diagnosis problem whose components are: (i) some new (super)components, each one representing the aggregation of a set of connected components of D, and (ii) the components of D which are not involved in the aggregations. Following the approach of the semantic theory of abstraction, we will first define a proper interpretation mapping, i.e., an interpretation mapping that encodes the aggregation of structure and behavior of components at the model level, and then define the structural abstraction of a diagnosis problem with respect to that interpretation mapping.

As we will see, structural abstraction is a MI abstraction. This will allow us to formalize a correspondence between the solutions of a diagnosis problem and the solutions of its structural abstraction. This correspondence will be the basis for the exploitation of structural abstraction in diagnostic reasoning.

At the end of this section, we include a comparison with previous formalizations of structural abstraction and highlight relations and improvements.

3.1. Informal definitions and assumptions

Suppose we are given a diagnosis problem D_0, and we want to build a more abstract version of it D_1, in which a single (super)component, called *sc*, replaces a set *AGGR* of connected components of D_0.

A first desirable requirement is that the parts of D_0 which do not refer to the components in *AGGR* remain identical in D_1. Thus, in general, we want that:

- $COMPS_1$ is composed by *sc* and those components of $COMPS_0$ that do not belong to *AGGR*;
- the behavioral descriptions BD_1 of D_1 is composed by the behavioral descriptions (which are the same as in D_0) of the components that do not belong to *AGGR*, and a behavioral description of *sc* (which must be provided);
- the compositional description CD_1 of D_1 connects the components that do not belong to *AGGR* as they are connected in CD_0 of D_0;
- OBS_1 contains those observations in OBS_0 that refer to components that do not belong to *AGGR*.

Moreover, we want to define a relation between the representation of *sc* and the representation of the components in *AGGR*, which can be then exploited in diagnosis (e.g., knowing that *sc* is faulty in some way, we want to make hypotheses on the faults in its subcomponents). We break down this requirement into three parts, which separately consider the behavioral description, the structural description, and the observations.

With respect to the behavioral description, we want each behavioral mode of *sc* to correspond to (more precisely, to be an abstraction of) a set of combinations of behavioral modes of its subcomponents. This denotes the fact that, in real systems, we can know what a supercomponent is doing by checking what the subcomponents are doing, and then composing their behaviors.

The fact that the components in *AGGR* are behaving in a certain way can be expressed by the formula $\bigwedge_{x \in AGGR} m_i(x)$ (where each m_i is a proper mode of the behavioral description of x), i.e., a partial candidate for the components in *AGGR*. The set of all possible combinations of behaviors for the components belonging to *AGGR* is the set $CANDS(D_0, AGGR)$. We then create a number of sets of partial candidates $BM_1, \ldots, BM_l$ that partition the set $CANDS(D_0, AGGR)$, in such a way that each BM_i corresponds to a behavioral mode for the supercomponent, i.e., the supercomponent is in a certain behavioral mode only if its subcomponents behave consistently with at least one of the partial candidates in the corresponding BM_i set.

From the structural point of view, since the supercomponent is supposed to replace the components in *AGGR*, the ports of the supercomponent will be all and only the ports of the components in *AGGR* that connect them to other components not in *AGGR*, i.e., the ports $\vec{z}_{sc}$ of the subsystem whose compositional description is

$$T(sc, \vec{z}_{sc}) \equiv \bigwedge_{c \in AGGR} T_c(c, \vec{z}_c)$$

where T will be the type of supercomponent *sc*. The ports that are internal to the subsystem which is aggregated will be then neglected in the abstraction.

Finally, since the internal ports of the subsystem which is aggregated are not present in D_1, observations concerning them will not be included in OBS_1.

3.2. Formal definitions

We now formally express the above illustrated requirements using the semantic theory of Nayak and Levy, i.e., using interpretation mappings to perform the aggregation at the model level. We begin by defining a basic structural abstraction that aggregates one set of connected components into a supercomponent, and then consider the general case, where more aggregations can be performed. The following definition characterizes the interpretation mappings for structural abstraction.

Definition 5 (*interpretation mapping for structural abstraction*). Let $D_0 = (SD_0, OBS_0, COMPS_0)$ be a diagnosis problem in language L_0. Let $SA = \langle AGGR, BM_1, \ldots, BM_k \rangle$ be a tuple of non-empty sets such that

- $AGGR \subseteq COMPS_0$ and $|AGGR| = k$,
- the sets BM_i are a partition of the set $CANDS(D_0, AGGR)$.

Let L_1 be the language that will be used to define the abstract diagnosis problem. L_1 includes *sc*, the name of the supercomponent that will result from the aggregation of the components in $AGGR$, T_{sc}, its type predicate, and $m_1, \ldots, m_k$ the predicates identifying its behavioral modes.

Let $\pi_{SA} : Models(D_0) \rightarrow Interpretations(L_1)$ be an interpretation mapping for which $\pi_{SA_\forall} \equiv (x_1 = x_1)$. π is an interpretation mapping for structural abstraction of D_0 if:

- $T_{sc}(sc, \vec{z}_{sc}) \equiv_{\pi_{SA}} \exists w_1, \ldots, w_n (\bigwedge_{c \in AGGR} T_c(c, \vec{z}_c))$,
- $m_i(sc, \vec{z}) \equiv_{\pi_{SA}} \bigvee_{pc \in BM_i} pc$, $i = 1, \ldots, k$,
- in any other case, π_{SA} is the identity function.

The interpretation mapping π_{SA} does not change the domains in which the system operates[3] (e.g., real numbers for flows in the hydraulic domain, or binary quantities for electronic systems).

The first requirement states that the denotation of the type predicate of *sc* is the restriction, in the abstract universe, of the compositional description of the subsystem that is aggregated. This establishes a link between a supercomponent and its underlying structure.

The second requirement states that the denotation of a given behavioral mode of *sc* is the restriction in the abstract universe of the disjunction of the partial candidates contained

[3] Changing them would require a behavioral abstraction.

in a corresponding *BM* set. This establishes a link between a set of partial candidates for the subcomponents, and the behavioral mode of the supercomponent that abstracts them.

Finally, the last requirement says that the denotation of any other object does not change, i.e., all the parts of D_0 which are not involved in the aggregation remain as they are.

As one can notice, such interpretation mapping "encodes" into its definition the sets $AGGR, BM_1, \ldots, BM_k$ which are the choices one makes when performing a structural abstraction in a diagnosis problem.

Given the definition above, we now define a basic structural abstraction of a diagnosis problem.

Definition 6 (*structural abstraction of a diagnosis problem—basic case*). Let $D_0 = (SD_0, OBS_0, COMPS_0)$ and $D_1 = (SD_1, OBS_1, COMPS_1)$ be two diagnosis problems in language L_0 and L_1, respectively, and $\pi_{SA}: Models(D_0) \rightarrow Interpretations(L_1)$ an interpretation mapping for structural abstraction of D_0. D_1 is a basic structural abstraction of D_0 with respect to π_{SA} if

- $COMPS_1 = (COMPS_0 - AGGR) \cup \{sc\}$,
- $CD_1 = T(S, \vec{z}_s) \equiv \exists z_1, \ldots, z_n (T_{sc}(sc, \vec{z}_{sc}) \wedge \bigwedge_{c \notin AGGR} T_c(c, \vec{z}_c))$ where the type declarations for the components not belonging to *AGGR* are as in CD_0,
- $BD_1 = (BD_0 - \bigcup_{c \in AGGR} BD_c) \cup BD_{sc}$,
- OBS_1 contains only those observations in OBS_0 that refer to ports that are present in D_1.

This definition specifies how the formulas of D_1 should be constructed. D_1 contains *sc* and all the components of D_0 which are not involved in the aggregation. The abstract compositional description CD_1 is obtained by simply removing from CD_0 the predicates that refer to the components in *AGGR*, and introducing the predicate for *sc*. Those ports that are internal to the subsystem *AGGR* are thus not included in D_1. The abstract behavioral description BD_1 contains the behavioral descriptions in BD_0 of the components that are not in *AGGR*, and the behavioral description of *sc*. The behavioral description of *sc* must satisfy the adopted interpretation mapping π_{SA}, i.e., the relation between behavioral modes and BM_i sets encoded in π_{SA}. Finally, OBS_1 is simply a subset of OBS_0, omitting the observations concerning ports that are not present in D_1.

Example 3. Consider the diagnosis problem illustrated in Example 1, and suppose we want to build a structural abstraction by aggregating the components *pm* and p_1 into a single supercomponent *sc*, i.e., $AGGR = \{pm, p_1\}$. We now define the sets $BM_1, \ldots, BM_k$ which will correspond to the behavioral modes of *sc*. Since there are four behavioral modes for the pump and two behavioral modes for the pipe, there are eight possible partial candidates for *pm* and p_1, which we group in the following four sets:

$$BM_1 = \{ok(pm) \wedge ok(p_1)\},$$
$$BM_2 = \{ok(pm) \wedge leak(p_1), leak(pm) \wedge ok(p_1), leak(pm) \wedge leak(p_1), loF(pm) \wedge leak(p_1), hiF(pm) \wedge leak(p_1)\},$$

$$BM_3 = \{loF(pm) \wedge ok(p_1)\},$$
$$BM_4 = \{hiF(pm) \wedge ok(p_1)\}$$

The first set represents a normal behavior; the second set represents all situations where there is a leak; the third and fourth sets correspond to situations where p_1 is normal, and *pm* is delivering a wrong amount of flow (low or high, respectively). The four sets will be abstracted by the behavioral modes of the supercomponent called *ok*, *leak*, *loF*, and *hiF*, respectively. By Definition 5, we construct an interpretation mapping π in which:

$$ok(sc) \equiv_\pi BM_1,$$
$$leak(sc) \equiv_\pi BM_2,$$
$$loF(sc) \equiv_\pi BM_3,$$
$$hiF(sc) \equiv_\pi BM_4,$$
$$TYPE(sc, [t_1, t_3]) \equiv_\pi \exists t_2 \left[pump(pm, [t_1, t_2]) \wedge pipe(p_1, [t_2, t_3])\right]$$

In the resulting abstract diagnosis problem, $COMPS = \{sc, n_1, p_2, p_3, v_1, v_2, p_4, p_5, n_2, p_6\}$; the behavioral description is the same given in Example 1, except that the behavioral descriptions for *pm* and p_1 are omitted, and the behavioral description for *sc* is added. Given the adopted interpretation mapping, the type *pump* can be assigned to *sc*. Finally, the abstract compositional description is

$$hydSystem(hcs, [t_1, t_{13}, s_{v_1}, s_{v_2}]) \equiv \exists t_3, \ldots, t_{12} \left[pump(sc, [t_1, t_3)]) \wedge \right.$$
$$twoOut(n_1, [t_3, t_4, t_5]) \wedge pipe(p_2, [t_4, t_6]) \wedge pipe(p_3, [t_5, t_7]) \wedge$$
$$valve(v_1, [s_{v_1}, t_6, t_8]) \wedge valve(v_2, [s_{v_2}, t_7, t_9]) \wedge pipe(p_4, [t_8, t_{10}]) \wedge$$
$$\left. pipe(p_5, [t_9, t_{11}]) \wedge twoIn(n_2, [t_{10}, t_{11}, t_{12}]) \wedge pipe(p_6, [t_{12}, t_{13}])\right]$$

We now define the general case for structural abstraction, where more than one basic structural abstractions are applied in order to obtain the desired level of structural complexity.

Definition 7 (*structural abstraction—general case*). Let D_0 be a diagnosis problem, D_1 and D_2 be (basic) structural abstractions of D_0 and D_1 with interpretation mappings π_A and π_B, respectively. Then, D_2 is a structural abstraction of D_0 with interpretation mapping $\pi_B \circ \pi_A$.

Typically, one might not want to directly abstract a system with many components into a system with a few or even one component, but rather wants to have more than one different structural abstractions of the same system, for example to be able to choose a proper level of detail for reasoning on the situation at hand.

This is usually accomplished by building a multi-level hierarchy of structural abstractions, i.e., a list of diagnosis problems, each one representing the same system at a different level of detail, with the first one (i.e., the bottom level of the hierarchy) being the most detailed one, and every other (i.e., the other levels of the hierarchy) being a structural abstraction of the previous one.

Definition 8 (*multi-level hierarchy of structural abstractions*). A multi-level hierarchy of structural abstractions H is an ordered set of q diagnosis problems $\{(SD_i, OBS_i, COMPS_i)\}$ with $i = 0, \ldots, q-1$, where for every j, with $j = 0, \ldots, q-2$, D_{j+1} is a structural abstraction of D_j with respect to some interpretation mapping π_j.

Example 4. A multi-level hierarchy of structural abstractions organized in six levels for the *hcs* is shown in Fig. 2. For each level, the figure illustrates the layout of the components and the ports, and highlights the components which have been aggregated.

In particular, in level 1 we aggregated pipe p_2 and valve v_1 into valve sc_1, and pipe p_3 and valve v_2 into valve sc_2. Both aggregations exploit a similar interpretation mapping: in the case of sc_1, the interpretation mapping π is such that (we omit the definitions of the BM_i sets, which can be derived from the right-hand part of the following sentences):

$$ok(sc_1) \equiv_\pi ok(p_2) \wedge ok(v_1),$$
$$leak(sc_1) \equiv_\pi leak(p_2) \vee leak(v_1),$$
$$stuckO(sc_1) \equiv_\pi ok(p_2) \wedge stuckO(v_1),$$
$$stuckC(sc_1) \equiv_\pi ok(p_2) \wedge stuckC(v_1)$$

and $TYPE(sc_1, [s_{v_1}, t_4, t_8]) \equiv_\pi \exists t_6[pipe(p_2, [t_4, t_6]) \wedge valve(v_1, [t_6, t_8])]$. In the case of sc_2, the interpretation mapping is the same, but the involved components are p_3 and v_2.

In level 2, we aggregated valve sc_1 and pipe p_4 into valve sc_3, and valve sc_2 and pipe p_5 into valve sc_4. Both aggregations are performed using an interpretation mapping analogous to the ones used in the previous level.

In level 3, we aggregated valves sc_3, sc_4 and three-way nodes n_1 and n_2 into sc_5, which is a valve with two ports for setting the state (sc_5 is closed if and only if both s_{v_1} and s_{v_2} are set to closed), whose component type predicate is *valve*2.

Its behavioral description is as follows:

$$valve2\big(x, [s_1, s_2, in, out]\big) \supset \big[ok(x) \supset ((s_1 = closed \wedge s_2 = closed \wedge in = out = 0) \vee ((s_1 = open \vee s_2 = open) \wedge in = out))\big],$$
$$valve2\big(x, [s_1, s_2, in, out]\big) \supset \big[leak(x) \supset in > out\big],$$
$$valve2\big(x, [s_1, s_2, in, out]\big) \supset \big[stuckO(x) \supset ((s_1 = closed \wedge s_2 = closed) \wedge in = out > 0)\big],$$
$$valve2\big(x, [s_1, s_2, in, out]\big) \supset \big[stuckC(x) \supset ((s_1 = open \vee s_2 = open) \wedge in = out = 0)\big]$$

In level 4, we aggregated *pm* and p_1 into pump sc_6 (using an interpretation mapping analogous to the one used in Example 3), and sc_5 and p_6 into sc_7 (which is a valve with two ports for the state as sc_5).

Finally, in level 5 we aggregated sc_6 and sc_7 into sc_8, which is basically a pump with two ports s_{v_1} and s_{v_2} for setting its state, and that delivers a constant flow equal to F_K when at least one port is set to open, and delivers no flow otherwise.

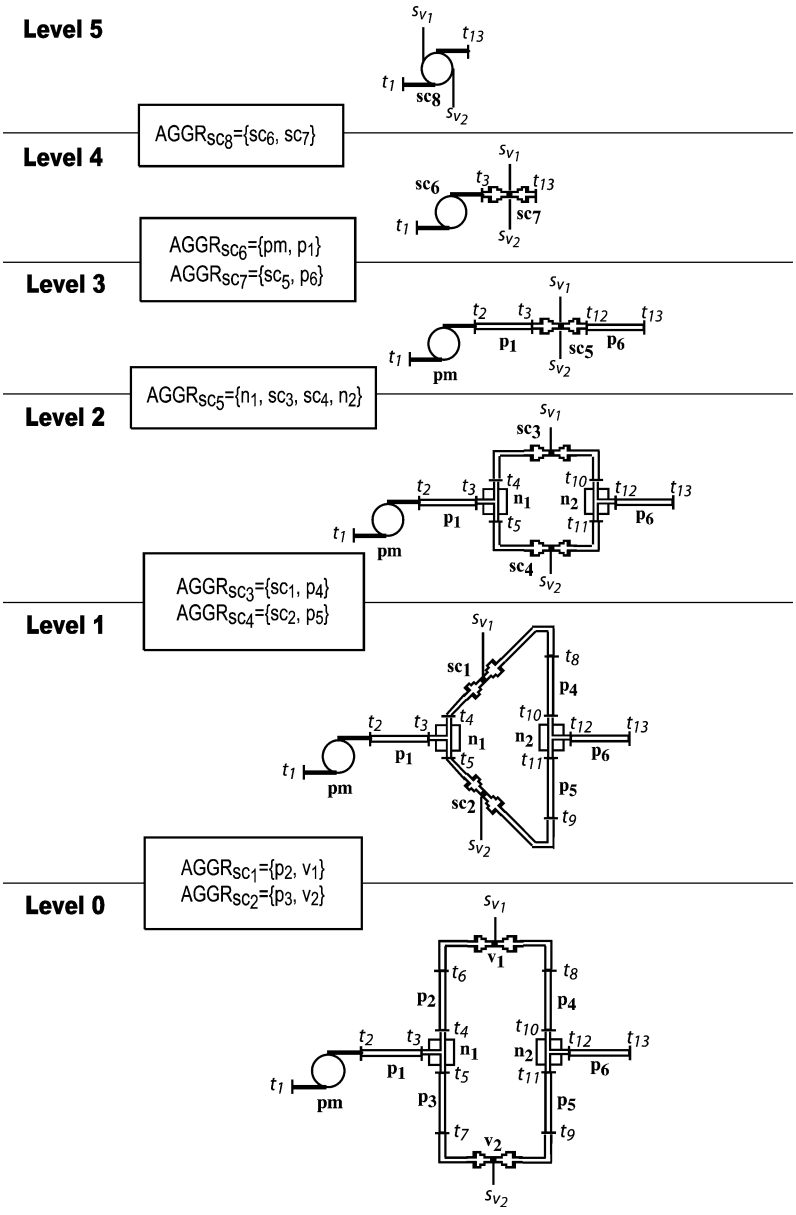

Fig. 2. A hierarchy of structural abstractions of the *hcs* with 6 levels.

3.3. *Properties of structural abstractions*

In this section, we examine the properties of structural abstractions that are useful in diagnosis. In particular, given a diagnosis problem D_0 and its structural abstraction D_1,

we first define the *refinement* of a candidate of D_1 as any candidate of D_0 which makes consistent predictions at a more detailed level. Then, we prove that structural abstractions are MI abstractions, and consequently have the properties illustrated in Section 2. Using these properties, we can prove that if a candidate of any structural abstraction of a diagnosis problem D is not a diagnosis, then all its refinements are not diagnoses of D.

Definition 9 (*refinement of a candidate*). Let D_0, D_1 be two diagnosis problems in languages L_0 and L_1, respectively, such that D_1 is a structural abstraction of D_0 with respect to an interpretation mapping π. Let *AGGR* be the set of components of D_0 that are aggregated, and let $sc \in COMPS_1$ be the name of their supercomponent.

Let $C_1 = PC \wedge m_{sc}(sc)$ be a candidate of D_1, where PC is a partial candidate for the components in the set $COMPS_1 - \{sc\}$. A candidate C_0 of D_0 is called a refinement of C_1 if:

- $C_0 = \bigwedge_{c \in AGGR} m_c(c) \wedge PC$,
- $m_{sc}(sc) \equiv_\pi \bigwedge_{c \in AGGR} m_c(c) \vee \Delta$, where Δ is a (possibly empty) disjunction of partial candidates for the components belonging to *AGGR*.

The idea is that a candidate and its refinements contain the same behavioral modes assignments for the components that are not involved in the abstraction, while the behavioral mode m for the supercomponent is substituted in the refinement with a partial candidate abstracted by m itself. Note that by definition of structural abstraction every candidate in an abstract diagnosis problem has at least one refinement (the BM_i sets are a partition of the set of all partial candidates, and then each partial candidate is abstracted into a behavioral mode for the supercomponent).

The following theorem states that every structural abstraction is also a MI abstraction.

Theorem 3. *Let D_0, D_1 be two diagnosis problems in languages L_0 and L_1, respectively, such that D_1 is a structural abstraction of D_0 with respect to an interpretation mapping π. Then, $D_1 <_\pi D_0$, i.e., D_1 is a MI abstraction of D_0 with respect to π.*

Proof (*outline*). Consider any model of D_0, which is an assignment I of values to the ports of D_0 for which there is at least a candidate C_0 of D_0 such that

$$I \models SD_0 \wedge OBS_0 \wedge C_0 \tag{1}$$

We must show that $\pi(I)$ is a model of D_1, i.e., there is a candidate C_1 of D_1 such that

$$\pi(I) \models SD_1 \wedge OBS_1 \wedge C_1 \tag{2}$$

By definition of structural abstraction, $\pi(I)$ and OBS_1 are restrictions of I and OBS_0, respectively, which assign a value only to the ports that are present in both D_0 and D_1. Thus,

$$\pi(I) \models SD_0 \wedge OBS_1 \wedge C_0 \tag{3}$$

One can substitute SD_0 with SD_1 in (3) because the denotation of CD_0 and CD_1, BD_0 and BD_1 are the same in the considered restriction of I. Then, we take C_1 such that C_0

is one of its refinements. The partial candidate contained in C_1 that assigns a mode to the components that are shared between D_0 and D_1 has $\pi(I)$ as a model (again, the restriction does not involve the components, observations, etc. that are not aggregated). Finally, by definition of refinement, $\pi(I)$ is a model also of the partial candidate contained in C_1 that assigns a mode to the aggregations of components in D_0. □

Since structural abstractions are MI, we can prove the following relation between any candidate of the more abstract diagnosis problem and its refinements:

Theorem 4. *Let D_0, D_1 be two diagnosis problems in languages L_0 and L_1, respectively, such that D_1 is a structural abstraction of D_0 with respect to the interpretation mapping π. Let C_1 be a candidate of D_1 which is not a diagnosis. Then, any refinement C_0 of C_1 is not a diagnosis for D_0.*

Proof. Since C_1 is not a diagnosis, then $C_1 \wedge SD_1 \wedge OBS_1$ is inconsistent. Moreover, D_1 is a structural abstraction of D_0, thus $SD_1 <_\pi SD_0$ and $OBS_1 <_\pi OBS_0$. Finally, $C_1 <_\pi C_0$ by hypothesis. By compositionality of MI abstractions we can write

$$C_1 \wedge SD_1 \wedge OBS_1 <_\pi C_0 \wedge SD_0 \wedge OBS_0$$

Since $C_1 \wedge SD_1 \wedge OBS_1$ is inconsistent, then for property 1 also $C_0 \wedge SD_0 \wedge OBS_0$ is inconsistent, i.e., C_0 is not a diagnosis. □

This theorem allows one to exclude from the possible solutions of a diagnosis problem D every refinement of an impossible diagnosis. The advantage is that, by eliminating a candidate in a (potentially simpler) abstract diagnosis problem, one can possibly rule out many candidates in the original diagnosis problem.

On the other hand, if we find that an abstract candidate is a diagnosis, we cannot guarantee that its refinements are diagnoses too, as shown by the following example.

Example 5. Consider the diagnosis problem for the *hcs* given in Example 1. Suppose that there is only one observations specifying that flow at port t_2 is greater than zero, but below F_K. In the structural abstraction given in Example 3, port t_2 is not present, thus the candidate

$$ok(sc) \wedge ok(n_1) \wedge ok(p_2) \wedge ok(p_3) \wedge ok(v_1) \wedge ok(v_2) \wedge ok(p_4) \wedge ok(p_5) \wedge ok(n_2) \wedge ok(p_6)$$

is a diagnosis for the abstract diagnosis problem, while its refinement

$$ok(pm) \wedge ok(p_1) \wedge ok(n_1) \wedge ok(p_2) \wedge ok(p_3) \wedge ok(v_1) \wedge ok(v_2) \wedge ok(p_4) \wedge ok(p_5) \wedge ok(n_2) \wedge ok(p_6)$$

is not a diagnosis, since the considered situation can only be explained by *pm* delivering a low amount of flow.

3.4. Some remarks on the aggregation of components

In general, when aggregating a set of components $c_1, \ldots, c_k$ into a component *sc*, the number of behavioral modes for *sc* can be (at worst) $N = \prod_{i=1,\ldots,k} M_i$, where M_i is the number of behavioral modes for component c_i. The motivation is that, in principle, each combination of behavioral modes for the subcomponents could be represented as a distinct behavioral mode for the supercomponent. In this case, the complexity reduction we want to obtain with the abstraction vanishes because of the number of supercomponent modes that have to be considered: since N is exactly the number of partial candidates involving the subcomponents, the number of candidates remains equal to the more detailed diagnosis problem.

Therefore, one needs to find suitable aggregations of components that are able to reduce the complexity of reasoning. In some cases, those aggregations are easily found, because some combinations of behavioral modes for the subcomponents will not be distinguishable in terms of values at the supercomponent ports, and thus they will be naturally combined into the same behavioral mode. In Example 3, the combinations of subcomponent modes

$$hiflow(pm) \wedge leak(p_1),$$

$$ok(pm) \wedge leak(p_1)$$

may predict different values for port t_3. In the first case, flow at t_3 may be greater than F_K, while in the second case it must be less than F_K. However, both sentences predict that input flow of *sc* is greater than its output flow, and thus both of them can be abstracted by a *leak* mode. However, easy aggregations may not be possible in some systems: in this case, a possible solution is to forget some combination of modes in the abstraction. This solution can be adopted, for example, when a combination of modes yields a very complex behavior. In such situations, the structural abstraction is not MI (each model of the forgotten combination of behaviors has no corresponding model in the more abstract diagnosis problem). However, it can be viewed as a structural abstraction under the simplifying assumption that the forgotten combination of modes cannot occur. This has to be taken into account by diagnostic reasoning, which has to consider two cases: one in which the simplifying assumption holds (where the abstract diagnosis problem can be used), and one in which the simplifying assumption does not hold (where only the more detailed diagnosis problem can be used). The union of the solutions of the two cases is the solution of the original diagnosis problem.

3.5. Related work

Formalizations of structural abstraction have been proposed by Struss [19], Mozetic [14], and more recently, by Autio and Reiter [1]. Genesereth [10] was probably the first to advocate the use of structural abstraction as a way to reduce the computational cost of model-based diagnosis, and to recognize also that loss of information can lead to undiagnosability of the abstract representation.

In [14], structural abstraction is addressed as follows. The representation of a system is constituted by a mapping m between any state of the system (normal or faulty) to corresponding input-output observations:[4]

$$m : x \rightarrow y$$

where x is a state of the system (e.g., a normal state) and y is a tuple of input-output observations.

The refinement/abstraction operators allow one to start from a mapping m representing the system and to derive a more abstract mapping m' representing the same system with less detail. To relate the two representations, one needs a function h that maps between states in m and states in m', and between input-output values in m and m'. That is, $h(x', x)$ maps between a state x of m and a state x' of m' (and vice versa), and the same function $h(y', y)$ maps between input-output values y of m and input-output values y' of m' (and vice versa).

When the system is composed by components $c_1, \ldots, c_n$, the representation can be defined by a composition of mappings in the following way:

$$m(x, y) \leftarrow c_1(x_1, z_1), \ldots, c_n(x_n, z_n)$$

where $c_1, \ldots, c_n$ are the mappings that define the relation between the states of the components and their input-output observations. One can aggregate some components, e.g., $c_1, \ldots, c_k$ in the representation into a supercomponent c and obtain the following simpler system representation:

$$m'(x', y') \leftarrow c(x_c, z_c), c_{k+1}(x_{k+1}, z_{k+1}), \ldots, c_n(x_n, z_n)$$

By repeated applications of abstraction operators, one can obtain a multi-level hierarchy of representations of the system, each one at a different level of detail.

To ensure correctness in diagnostic reasoning, it must be guaranteed that the diagnoses which are impossible at an abstract level are impossible at a more detailed level as well. This requirement is formulated by Mozetic in terms of a global consistency condition that must be satisfied among the different levels: given two adjacent levels of the hierarchy (m more detailed, and m' more abstract), the consistency condition is expressed as:

$$\begin{aligned} &\forall x, y m(x, y) \wedge \left(\exists x'_1\, h(x'_1, x)\right) \\ &\quad \Rightarrow \exists x', y'\, m'(x', y') \wedge h(x', x) \wedge h(y', y) \end{aligned} \tag{4}$$

which guarantees that (see [14] for a detailed discussion):

- given a more detailed mapping m, a state x with an abstraction x' cannot be mapped to input-output values y which do not have a corresponding abstract y';
- if x maps to y (i.e., in state x the possible input-outputs are y), and we have abstractions x' and y' for x and y respectively, then x' must map to y'.

[4] We repeat here the original notation of [14]. Note that letters (such as m and c) may have different meanings from those adopted in the rest of this paper.

Our definition of structural abstraction (as shown by Theorem 4) satisfies the consistency condition. Our definition is slightly stronger in the sense that every part of the detailed representation must be abstracted, while Mozetic's definition allows for incompleteness in the abstraction process (e.g., some faulty modes could not be abstracted at all). However, as discussed in the previous section, we introduce that possibility by using simplifying assumptions.

As a consequence, the consistency condition in our approach needs not to be checked for each derived abstract level, but is enforced during the abstraction process (i.e., the interpretation mapping used by each aggregation must satisfy Definition 5). In other words, our definition clearly states the rules that each aggregation has to follow in order to guarantee the consistency condition, while in Mozetic's formalization the consistency condition has to be verified globally for each level.

In [19], structural abstraction is viewed as one of the possible *model relations* called *refinement*. Refinement defines structural abstraction for diagnosis problems as follows. Suppose that formula $ok(c)$ represents the correct behavioral mode of a supercomponent sc, and that formulae $ok(c_1), \ldots, ok(c_k)$ represent the correct behavioral modes of its subcomponents $c_1, \ldots, c_k$, respectively. Then, $ok(sc)$ (the theory describing the supercomponent's behavior) is a structural abstraction of $ok(c_1), \ldots, ok(c_k)$ (the theories describing the subcomponents' behavior) if

$$ok(c_1) \wedge \cdots \wedge ok(c_k) \Rightarrow ok(sc),$$

which expresses the assumption that "*If all parts of a system work correctly, then so does the entire system*". This formalization does not deal with components with multiple behavioral modes, so it is applicable only to some diagnosis problems. Moreover, the formalization characterizes only the behavioral descriptions of the supercomponent, without dealing at all with the representational changes in structure that are brought by structural abstraction. Our formalization can be seen as an extension of the one proposed by Struss: if one represents only the normal behavior of components, the requirement on our abstraction mapping becomes

$$ok(sc) \equiv_{\pi} \bigwedge_{c_i \in AGGR} ok(c_i)$$

which is the same definition of structural abstraction given by Struss, but formalized using the semantic theory of abstraction.

In the approach proposed by [1], a hierarchical representation is composed by multiple system descriptions, each one at a different level of structural detail, and a *component hierarchy*, which is represented as a tree with nodes corresponding to components at any level of detail. More precisely, the leaves are the components belonging to the most detailed description, while interior nodes are abstract components obtained by successive aggregations starting from the primitive components. The component hierarchy represents the structural abstractions that have been performed in order to build the hierarchical representation. Autio and Reiter [1] do not consider components with multiple behavioral modes, but diagnostic theories where the behavioral description consists only in the correct behavior of components.

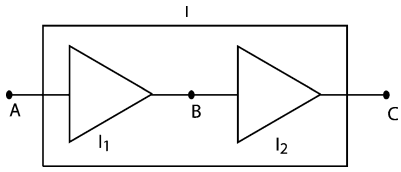

Fig. 3. Layout of a circuit with two inverters.

A system $(SD', COMPS')$ is defined to be a structural abstraction of a system $(SD, COMPS)$ if for every component c in $COMPS'$ there exists a system $(SD_c, COMPS_c)$ such that its components are components of $COMPS$, and these subsystems are properly connected to form the compositional description of $(SD, COMPS)$. It is required that the structural abstraction of a set of subcomponents $c_1, \ldots, c_m$ into their supercomponent c must satisfy the following assumptions:

- if a (super)component c is abnormal, then at least one of its subcomponents $c_1, \ldots, c_m$ must be abnormal as well, i.e., $\neg ok(c) \Rightarrow \neg ok(c_1) \vee \cdots \vee \neg ok(c_m)$,
- if a (super)component is normal, then all its subcomponents are normal as well, i.e., $ok(c) \Rightarrow ok(c_1) \wedge \cdots \wedge ok(c_m)$.

These assumptions are referred to by Autio and Reiter as *abnormality axioms*. The first abnormality axiom is equivalent to the definition by Struss discussed above. The second abnormality axiom poses serious problems in situations such as fault masking or fault compensation. For example, consider the well-known logic circuit shown in Fig. 3, composed by two inverters I_1 and I_2. Suppose that the measurements report $A = 1, B = 1, C = 1$, indicating that both inverters are faulty (e.g., they are both stuck at 1). We aggregate both inverters into a single component I with input A and output C, as shown by the figure. By looking only at I, we can consider it normal, because the two associated observations are not symptoms (this happens because the two faults in inverters I_1 and I_2 mask each other). By applying the second abnormality axiom, we conclude that both I_1 and I_2 are normal, which is not true. Thus, when fault masking occurs, the second abnormality axiom can result in wrong conclusions.

We critique the second abnormality axiom also from a common-sense point of view. It says that "if a system works correctly, then so do all its parts": an expert would use this assumption only to focus reasoning first on more evident faults, but then would remove it to consider more subtle faults.

Finally, Autio and Reiter prove that structural abstraction in their formalization is *sound*, in the sense that for every diagnosis D involving a supercomponent c, there is at least a corresponding diagnosis where c is substituted by its subcomponents. They also prove that structural abstraction is not *complete*, in the sense that there is the possibility of ruling out valid diagnoses when reasoning at abstract levels (this is shown also in the circuit example). This is a serious limitation in reasoning, because it means that one cannot exclude impossible diagnoses by reasoning at an abstract level, and thus, in order to find all possible diagnoses, one has to reason only with the most detailed representation available.

4. Reasoning with structural abstraction

In this section, we will examine the problem of diagnostic reasoning with structural abstraction. First, we will consider the well-known Mozetic's approach [14] to hierarchical diagnosis and apply it to multi-level hierarchies of structural abstractions. Then, we will analyze why, in many cases, the efficiency of Mozetic's approach is not as good as one would expect. In general, this originates from the fact that the same hierarchical representation is used for a system, regardless of the currently available observations. Finally, we will propose two novel approaches that, in the case of structural abstraction, overcome this problem, and we will show the obtained improvements using the experimental evaluation we performed.

4.1. A formalization of Mozetic's algorithm for multi-level hierarchies of structural abstractions

In the following, we provide a procedural formulation of Mozetic's approach for consistency-based diagnosis problems, clearly separating the main activities that have to be performed, namely abstraction of observations and fault diagnosis. Our formalization of Mozetic's algorithm will be called *Hierarchical Diagnosis* (*HD*) in the remaining part of the paper.

The key idea of *HD* is to apply (in principle) any plain, one-level diagnostic strategy to a multi-level hierarchical representation. The strategy is applied one level at a time, starting from the most abstract possible level and then proceeding to lower ones, and the diagnoses that are found at each level are used to reduce the number of possible diagnoses at lower levels.

In our formalization, the input given to *HD* is a multi-level hierarchy of q diagnosis problems $D_i = \{(SD_i, OBS_i, COMPS_i)\},\ i = 0, \ldots, q-1$. For simplicity, input observations are given at the most detailed level.[5]

We assume that two functions, *Abstract* and *Detailed*, which depend on the abstractions that have been employed to build the multi-level hierarchy, are available:

- $Abstract : OBS_j \rightarrow OBS_{j+1},\ j = 0, \ldots, q-2$, which corresponds to Mozetic's *abstract* predicate, mapping the observations at level j to observations at level $j+1$, and
- $Detailed : C_{j+1} \rightarrow C_j,\ j = 0, \ldots, q-2$, which corresponds to Mozetic's *detailed* predicate, mapping a candidate at level $j+1$ into its refinements at level j.

We formalize *HD* in Fig. 4, dividing it into two separate procedures. The first one (`Abstract-Observations`) associates the available observations to the proper abstract levels of the hierarchical representation. Then, the second one (`Top-Down-diagnosis`) performs the diagnosis.

[5] Giving observations at any level of the hierarchy would be straightforward: one need only to slightly modify the `Abstract-Observations` procedure in Fig. 4 such that the mapping of observations is performed both upwards and downwards.

```
Procedure Abstract-Observations
    l ← 0;
    q ← number of levels of the hierarchical representation;
    WHILE l < q − 1 ∧ OBS_l ≠ ∅ DO
        OBS_{l+1} ← Abstract(OBS_l);
        l ← l + 1;
    ENDWHILE;
    IF OBS_l = ∅ THEN CLO ← l − 1;
Procedure Top-Down-Diagnosis
    l ← CLO;
    Cand_l ← all candidates at level l;
    WHILE l ⩾ 0 DO
        Diag_l ← Verify(D_l, Cand_l);
        \* CandNA_l ← all candidates at level l which have no abstraction;
        IF l < CLO THEN
            Diag_l ← Diag_l ∪ Verify(D_l, CandNA_l); *\
        IF l > 0 THEN Cand_{l−1} ← Detailed(Diag_l);
        l ← l − 1;
    ENDWHILE;
HIERARCHICAL DIAGNOSIS
    Abstract-Observations;
    Top-down-Diagnosis.
```

Fig. 4. The *HD* algorithm (i.e., Mozetic's algorithm for multi-level hierarchies of consistency-based diagnosis problems). Statements in comments have to be executed only when using structural abstractions with simplifying assumptions.

In particular, `Abstract-Observations` takes as input the initially available observations OBS_0 (which refer to level 0 of the hierarchical representation), and, by applying the *Abstract* function, determines the available observations (if any) for level 1. The same function is then repeatedly applied to derive the observations for the other levels, until a level with no available observations is reached or all levels have been considered. The number of the Coarsest Level with Observations (hereinafter, *CLO*) is stored into variable *CLO*. Levels from *CLO* to 0 are then considered by procedure `Top-down-Diagnosis` in the following way:

- First, all the diagnoses that are consistent with the observations at the current level are determined by using the function *Verify* (corresponding to the *verify* predicate in Mozetic's original formulation). *Verify* takes the diagnosis problem D_l at the current level, and a set $Cand_l$ which contains the possible diagnoses at the current level (for level *CLO*, all candidates are considered), and returns those elements of $Cand_l$ which are diagnoses, i.e., every $C \in Cand_l$ such that $SD_l \cup OBS_l \cup C$ is consistent.[6] After the *Verify* function has been applied, $Diag_l$ stores the diagnoses at the current level.

[6] It must be noted that no assumption is made on the implementation of this function (e.g., it can exploit a generate-and-test method, a GDE-like reasoner [7,9], or other model-based diagnosis techniques).

- Second, if l is greater than zero (i.e., current the level is not the most detailed one), the algorithm applies the function *Detailed* to the obtained diagnoses in order to derive their more detailed versions at level $l-1$, which will be the possible diagnoses at level $l-1$.

After level 0 has been considered, the `Top-Down-Diagnosis` procedure ends, and the final diagnoses are found in variable $Diag_0$.

If some candidates of the currently considered level have no abstraction at (previously considered) more abstract levels (i.e., we are using some simplifying assumptions), then they must be considered at the current level for the first time. In such case, one has to execute also the statements provided in comments in Fig. 4: in that way, the algorithm applies a second time the *Verify* procedure, but considering only candidates with no abstraction (stored in the variable $CandNA_l$) as possible diagnoses. All found diagnoses are then detailed into candidates for the next level.

The correctness and completeness of the algorithm rely on the *consistency conditions* discussed in the previous section. Since structural abstraction satisfies them, we can apply *HD* to any multi-level hierarchy of structural abstractions without having to check any consistency condition.

By taking advantage of the smaller search space at the more abstract levels, in many cases *HD* is able to outperform diagnostic reasoning applied only to the most detailed level. From a theoretical point of view, the speed up that can be achieved in the ideal case is exponential [14]. Experimental testing performed by Mozetic reported, in one medical example involving a four-level qualitative representation of the heart, a speed up by a factor of 20 over one-level diagnosis. Our experimental results with *HD* confirm similar improvements in the average case (as illustrated in Section 4.4).

4.2. Some remarks on the efficiency of HD

In this section, we use two examples to show that the efficiency of *HD* depends on both the chosen multi-level hierarchical representation and the actual observations that are given as input.

Example 6. Using the six-level hierarchical representation of the *hcs* illustrated in Example 4 and Fig. 2, we suppose that the following observation is available: flow at port t_7 has a value that is different from expected. Using the chosen multi-level representation, this observation cannot be abstracted to level 1, because t_7 is not present at that level. Hence, at the end of the execution of the `Abstract-Observations` procedure, $CLO = 0$. Since diagnosis is started at level 0, the efficiency of *HD* is similar (slightly worse, due to the extra reasoning activities of the `Abstract-Observations` procedure) to direct diagnosis of the most detailed representation of the system.

Note that by using a different multi-level representation, we can obtain better results with *HD* in Example 6. Consider an alternative six-level representation of the *hcs* which differs from the one illustrated in Fig. 2 only in levels 1 and 2: in level 1, sc_2 is obtained by aggregating v_2 and p_5 into sc_2 (instead of p_3 and v_2), and in level 2, sc_4 is obtained by

aggregating p_3 and sc_2 (instead of sc_2 and p_5). In this case, the observation of Example 6 can be abstracted up to level 1, and diagnosis would start from a more abstract problem ($CLO = 1$).

Example 7. Using the six-level hierarchical representation of the *hcs* illustrated in Example 4 and Fig. 2, we suppose that the following observations are available: flow at the port t_6 is $a > 0$, and valve v_1 is expected to be closed (i.e., $s_{v_1} = closed$). Using the chosen multi-level representation, the observation on t_6 cannot be abstracted to level 1, but the observation on s_{v_1} can be abstracted up to level 5. Hence, *CLO* is equal to 5. Unfortunately, at this level, the observation on s_{v_1} allows one to exclude only those candidates in which v_1 is stuck at open. Then, reasoning at levels 4, 3, 2, and 1 does not allow to exclude other candidates. At each of these levels, diagnostic reasoning has to consider a lot of candidates without useful results (and thus using the hierarchical algorithm is counterproductive in terms of efficiency).

More generally, the situations that can heavily limit the effectiveness of *HD* are the following:

- the algorithm cannot use at all some abstract levels (where potentially a lot of candidates could be excluded with little effort), since no observations are available for them (as Example 6 shows);
- only a few of the given observations are available at those abstract levels where diagnosis can be performed. In this case, diagnosing those levels could be scarcely effective in reducing the candidate space, or even be counterproductive, resulting in a loss of efficiency (as Example 7 shows).

With structural abstraction, when a component is aggregated all the observations that refer only to it become unavailable at the upper levels. Therefore, to avoid the above mentioned situations, one should build the multi-level representation in such a way that the relevant diagnostic information (i.e., current observations) is kept as much as possible available at more abstract levels, i.e., diagnosability at abstract levels is preserved. In principle, this implies that for every set of observable ports, one should build a different multi-level representation to guarantee efficient hierarchical reasoning.

However, building a new hierarchical representation by hand for each possible diagnostic scenario cannot be a viable approach: multi-level system representations are often hard to build, and the only one that could be possibly readily available is the one coming from system design. Moreover, general automatic methods for building multi-level representations are not available, or are heavily dependent on a particular system or domain (e.g., [3,13]).

In a system where the position of measurement sensors is known and does not change, one can think about building only one multi-level representation by using sensors position to guide the abstraction process, i.e., choosing the aggregations in such a way that the most abstract levels have always (or at least often) some observation available. However, this solution: (i) is not applicable to every system and (ii) whenever some sensors do not return measurements (e.g., because they are broken) the chosen hierarchical representation may

not be effective anymore, because some abstract levels may not have sufficient observations available.

In our research, we do not aim at building the optimal multi-level hierarchy for a given set of observations, but instead consider any chosen multi-level hierarchical representation, and aim at preserving diagnosability at abstract levels in those situations where the given observations do not allow *HD* to be effective.

4.3. Extensions to Mozetic's approach

In the following, we propose two extensions of *HD* that tackle the problems highlighted in the previous section.

More specifically, the extensions are called *REARRANGE* (presented in Section 4.3.2) and *BOTTOM-UP* (presented in Section 4.3.3). The idea behind both extensions is to tailor an existing multi-level hierarchical representation in order to obtain a more efficient diagnosis with the current observations (to this purpose, each extension uses a different strategy).

Our approach does not exploit any specific system or domain knowledge, and thus it is not limited to a particular domain or system. Moreover, the two extensions can be easily combined together in order to sum up their respective advantages.

Both extensions exploit an additional data structure, called *structural tree*, which is presented in the next section.

4.3.1. The structural tree

We associate any multi-level hierarchical representation (built with structural abstractions) H to a data structure $ST(H)$, called a *structural tree*. In the structural tree, each node represents a component of some level of H, and the sons of the node correspond to the subcomponents. In the following, given a tree T, we use the function $root(T)$ with its obvious meaning; for each node $n \in T$, the function $sons(n)$ returns the set of its sons.

Given a multi-level hierarchical representation of a diagnostic problem $H = \{(SD_i, OBS_i, COMPS_i)\}$, $i = 0, \ldots, q-1$, the structural tree $ST(H)$ is built as follows:

- first, for each component $c \in COMPS_0$, a leaf c is created;
- then, for each aggregation of $AGGR \subseteq COMPS_i$ into $sc \in COMPS_{i+1}$ such that each $c \in AGGR$ has already been added as a node of the tree, a node sc is created such that $sons(sc) = AGGR$.

The root of $ST(H)$ is associated with the component that represents the whole system, i.e., $root(ST(H)) = c \in COMPS_{q-1}$.

In general, one might not have a most abstract level with only one component representing the whole system. In this case, the derivation procedure outlined above would not derive a tree but a forest. In the remaining part of the paper, we assume that the most abstract level always contains just one component. However, multi-level hierarchies where this is not true can be still represented by a structural tree by simply adding an additional abstract level with just one component (representing the whole system) having just one

behavioral mode which abstracts all the candidates of the original most abstract level. Obviously, the newly added level would not bring any efficiency advantage.

Since each node $c \in ST(H)$ is a component $c \in COMPS_i$ for some $i = 1, \ldots, q-1$, we associate to it the theory $SD_c \subseteq SD_i$ and the observations $OBS_c \subseteq OBS_i$ (with OBS_c possibly empty).

The ST of a multi-level hierarchical representation H can be used to derive new levels (that are not in H) using the same aggregations that were employed in H. The following proposition defines the conditions under which a set of nodes in the ST characterizes a diagnosis problem for the considered system.

Proposition 5. *Given a multi-level hierarchical representation H, any set S of nodes belonging to $ST(H)$ such that*:

- *for each node in the set, no descendants or ancestors are included in the set,*
- *the descendants of all nodes in the set include all the leaves*

characterizes a diagnosis problem for the considered system, in which $COMPS = S$.

The resulting diagnosis problem can be easily built by retrieving from H all the necessary behavioral descriptions and observations, and by deriving a proper compositional description by simple conjunction of the type predicates for the chosen components (the needed instances of the type predicates are taken from the compositional descriptions of the diagnosis problems in H).

Example 8. Consider the multi-level hierarchical representation of the *hcs* shown in Fig. 2. Fig. 5 shows its structural tree. The diagnosis problem with components sc_6, n_1, sc_3, sc_2, p_5, n_2, and p_6 is not in the multi-level representation, but is a diagnosis problem for the *hcs*.

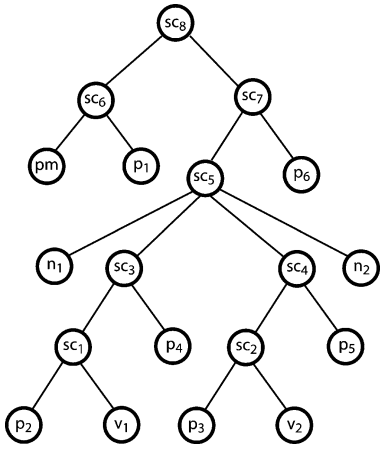

Fig. 5. Structural tree for the *hcs* derived from the multi-level hierarchical representation of Fig. 2.

4.3.2. The REARRANGE extension to Mozetic's approach

The extension presented in this section dynamically determines a multi-level hierarchical representation suited to diagnose the specific situation described by the current observations. The derived multi-level hierarchical representation will be built by *rearranging* the levels of a given hierarchical representation H by reasoning with the corresponding $ST(H)$ and the available observations.

The idea is to improve *HD* when it is not able to fully exploit the hierarchical representation, by providing a new, tailored hierarchical representation consisting of: (i) all levels of the original hierarchical representation which contain observations (i.e., the levels considered by *HD*), and (ii) additional, more abstract levels which are not present in the original hierarchy, and have the same observations of the coarsest level considered by *HD*.

We now show with an example how these additional levels can be built with little effort. Consider the scenario presented in Example 6, where diagnosis is started at level 0 (which contains 11 components). An hypothetical diagnosis problem (let us call it *Better*) with components sc_6, n_1, sc_3, p_3, v_2, n_2, p_5, and p_6 is more abstract than level 0 and has the same observations (i.e., flow at t_7 is not as expected). The search space of *Better* is much smaller: the diagnosis problem at level 0 has 4096 possible candidates, while *Better* has 512 possible candidates.

HD cannot start from *Better* because this diagnosis problem is not available in the given hierarchical representation: the only level with observations is level 0.

We can derive *Better* by properly selecting nodes in the structural tree of Fig. 5. The structural tree highlights indeed the aggregations used in the hierarchical representation, regardless of the order in which they were performed. One can derive *Better* by selecting a set of nodes in the structural tree which: (i) represent the full system, (ii) keep the same observations that are present at the level where *HD* starts, (iii) are as abstract as possible. This can be done by starting with a set of nodes which corresponds to *HD* starting level, and then try to substitute some of them with their parent provided that no observation is lost. Each possible substitution derives a new level (i.e., all requirements imposed by Proposition 5 are satisfied), from which the process of searching for more abstract levels can be repeated.

We implemented this strategy as an extension to *HD*, which we call *REARRANGE*. More specifically, the additional activity of building new levels is performed between the `Abstract-Observations` and the `Top-Down-Diagnosis` procedures, as shown in Fig. 6.

The added `Rearrange` procedure operates as follows:

- each observation available at level *CLO* (which is the same calculated by *HD*) is associated to the corresponding nodes in $ST(H)$ (obtaining the OBS_n sets);
- then, possible new levels, which are more abstract than *CLO*, are added to the levels of H with available observations, deriving a new multi-level hierarchical representation H'. These new levels are built by:
  (1) considering the components of the current level (which initially is *CLO*), which are assigned to the set *NodesToCheck*;

```
Procedure HIERARCHICAL-DIAGNOSIS
  Abstract-Observations;
  Rearrange;
  Top-down-diagnosis.

Procedure Rearrange
  IF CLO < g − 1 THEN
    H′ ← all levels of H from 0 to CLO;
    FOR each node n of ST(H) referred by OBS_CLO
      OBS_n ← {a = v|a = v ∈ OBS_CLO ∧ a is a port of n};
    NodesToCheck ← COMPS_CLO;
    WHILE NodesToCheck ≠ ∅ DO
      n ← first(NodesToCheck);
      c ← father(n);
      IF sons(c) ⊆ COMPS_CLO ∧ (⋃_{s∈sons(c)} OBS_s) = OBS_c THEN
          CLO ← CLO + 1;
          COMPS_CLO = (COMPS_CLO−1 − sons(c)) ∪ {c};
          ADD a new level numbered CLO with components COMPS_CLO to H′;
          NodesToCheck ← COMPS_CLO;
      ELSE
          NodesToCheck ← NodesToCheck − sons(c);
      ENDIF;
    ENDWHILE;
  ENDIF.
```

Fig. 6. The *REARRANGE* extension to *HD*.

(2) finding a subset of *NodesToCheck* that contains all and only the sons of a node c in $ST(H)$ such that the observations associated to the sons are exactly the observations associated to c (i.e., by substituting the sons with their father we do not lose any observation);
(3) if such set of components does not exist, the algorithm stops; otherwise, we add to H' a new top level obtained from the current level by substituting the sons of c with c itself. The newly added level is a diagnosis problem for the considered system, since it satisfies the requirements of Proposition 5. Since it is required that no observation is lost in the substitution, the newly derived level contains exactly the same observations as the starting one. Then, we go to point 2 considering the newly added level as the current one.

After the execution of the `Rearrange` procedure, the most abstract level derived by it is the new top level from which diagnosis is started. Note that when `Abstract-Observations` is already able to reach the most abstract level in H, then the `Rearrange` procedure does nothing, and the whole reasoning process is identical to *HD*.

The computational cost of the `Rearrange` procedure is polynomial in the number of nodes of the structural tree: for each newly generated level, at worst all the nodes in the level are examined once, and each examination consists of testing whether the node has

Table 2
Levels employed by *HD* (HD) and *REARRANGE* (R) in Example 9. The last two columns indicate whether the corresponding approach is able or not to exploit the level for top-down diagnosis

Level	Components	Candidates	HD	R
3	sc_6, n_1, sc_3, p_3, v_2, p_5, n_2, p_6 (8)	512	no	yes
2	*pm*, p_1, n_1, sc_3, p_3, v_2, p_5, n_2, p_6 (9)	1024	no	yes
1	*pm*, p_1, n_1, p_3, sc_1, v_2, p_4, p_5, n_2, p_6 (10)	2048	no	yes
0	*pm*, p_1, n_1, p_3, p_2, v_1, v_2, p_4, p_5, n_2, p_6 (11)	4096	yes	yes

a father, whether the observations associated to the father are the same of its sons, and whether all its sons belong to the current level.

We now reconsider the previously illustrated examples, and show how the *REARRANGE* extension behaves on them.

Example 9. Considering the observations given for the *hcs* in Example 6 (i.e., flow at port t_7 is not as expected), the *CLO* in the original multi-level hierarchy is level 0. The `Rearrange` procedure is able to derive first a level 1 with components *pm*, p_1, n_1, sc_1, p_3, v_2, p_4, p_5, n_2, and p_6 (p_2 and v_1 can be aggregated into sc_1 without losing observations). Then, level 2 is derived with components *pm*, p_1, n_1, sc_3, p_3, v_2, p_5, n_2, and p_6 (sc_1 and p_4 can be aggregated into sc_3 without losing observations). Finally, the most abstract level (from which diagnosis will start) is derived with components sc_6, n_1, sc_3, p_3, v_2, p_5, n_2 and p_6 (*pm* and p_1 can be aggregated into sc_6 without losing observations). The number of candidates associated to the above candidates levels are shown in Table 2.

Example 10. Considering the observations given for the *hcs* in Example 7 (i.e., flow at port t_6 is $a > 0$, and valve v_1 is expected to be closed), since the abstractions of observations proceeds up to level 5, the `Rearrange` procedure is not executed (because *CLO* is the most abstract level of the hierarchy), and the diagnostic reasoning proceeds exactly as in *HD*.

4.3.3. The BOTTOM-UP extension to Mozetic's approach

In this section, we propose a different strategy that also uses the available observations to reduce the complexity of hierarchical diagnosis. This second extension (called *BOTTOM-UP*) tries to exploit at a given level the observations that are only available at more detailed levels, and were forgotten in the abstraction process. The additional diagnostic information derived is used to eliminate impossible diagnoses, thus reducing the computational cost of reasoning at more detailed levels.

We now show with an example how the strategy works. Consider the scenario presented in Example 7, where diagnosis is started by *HD* at level 5. The only information available at levels 5, 4, 3, 2, and 1 is $s_{v_1} = \mathit{closed}$, which only allows one to eliminate all candidates in which $s_{v_1} = \mathit{open}$. However, a *stuckO* fault in valve v_1 is easily detectable at level 0 (since flow at the input of the valve is greater than zero), by running a diagnose procedure that separately considers only the observed components (i.e., p_2 and v_1). Thus, all the behavioral modes of valve v_1, with the exception of *stuckO*, can be removed from its behavioral description.

Moreover, since the *ok* mode is excluded from v_1, we can also remove all the behavioral modes that are abstractions of *ok* in all its supercomponents. In this way, we can exclude also $ok(sc_1)$, $ok(sc_3)$, $ok(sc_5)$, $ok(sc_7)$, $ok(sc_8)$. As a result, when reasoning starts at level 5, we already know that the system cannot be normal.

The *BOTTOM-UP* strategy extends *HD* by executing, after the abstraction of observations, a reasoning activity that considers all observed nodes in the ST, and removes from the behavioral representation associated to the node every mode which is not consistent with the associated observations.

Moreover, since the additional reasoning activity is performed using the *ST* in a bottom-up fashion, the diagnostic conclusions reached for the subcomponents can be reused when considering a node: in particular, since each mode m of a component c abstracts a set of partial candidates *BM* for the subcomponents of c, if all partial candidates in *BM* have been previously removed we can remove also m (even if the observations associated to c in isolation would not allow to do it).

The *BOTTOM-UP* extension to *HD* is illustrated in Fig. 7. The new activities performed by the `Bottom-up` procedure are:

(1) building of a list (recorded into the variable *NodesToCheck*) of the observed nodes in the structural tree, and all their ancestors; these nodes are then visited in increasing level order (i.e., when a node is visited, all its sons have been already visited);
(1) for each node c in the list, we consider each mode m of its behavioral description BD_c, and try to eliminate m in the following way:
- first, we check if m is actually an abstraction of previously eliminated partial candidates for the subcomponents of c: we determine with the function *Detailed*

```
Procedure HIERARCHICAL-DIAGNOSIS
  Abstract-Observations;
  Bottom-up;
  Top-Down-Diagnosis.

Procedure Bottom-up
  IF CLO ≠ 0 THEN
    NodesToCheck ← ∅;
    FOR each node n of ST(H) referred in OBS_0
       NodesToCheck ← NodesToCheck ∪ {n} ∪ ancestors(n);
    order NodesToCheck by increasing level;
    WHILE NodesToCheck ≠ ∅ DO
        n ← first(NodesToCheck);
        FOR each mode m of n
            IF (Detailed(m(n)) = ∅) ∨
                  (OBS_n ≠ ∅ ∧ Verify((BD_n, OBS_n, {n}), {m(n)}) = ∅) THEN
                  remove m from BD_n;
        remove n from NodesToCheck;
    ENDWHILE;
  ENDIF.
```

Fig. 7. The *BOTTOM-UP* extension to Mozetic's algorithm.

(defined previously) the partial candidates abstracted by m, and if the result is the empty set (because previous reasoning on the sons of c eliminated all those candidates), then we remove m from the behavioral description of c;

- if *Detailed*$(m(c))$ is not empty, then using the function *Verify* (defined previously), we check the consistency of candidate $m(c)$ with the observations on c (i.e., OBS_c) and the behavioral description of c (i.e., BD_c). If *Verify* does not return the partial candidate $m(c)$, then mode m is not consistent with the observations, and is removed from the behavioral description of c.

The computational cost of the `Bottom-up` procedure can be characterized as follows. At worst, all nodes of the *ST* have to be considered. For each node, we have to solve a diagnosis problem with just one component (which requires to verify a small number of modes and requires no propagation of values). Note that some modes in the supercomponent may require no verification because all their possible refinements have been excluded by reasoning with subcomponents. However, in a worst case scenario the computational complexity of the `Bottom-up` procedure can be characterized as the sum of the costs required to diagnose each node of the *ST* in isolation.

We now reconsider the previously illustrated examples, and show how the *BOTTOM-UP* extension behaves on them.

Example 11. Considering the observations given for the *hcs* in Example 6 (i.e., flow at port t_7 is not as expected), the *CLO* in the original multi-level hierarchy is level 0. Since the most abstract level with observations is level 0, the `Bottom-up` procedure is not executed, and reasoning proceeds exactly as in *HD*.

Example 12. Considering the observations given for the *hcs* in Example 7 (i.e., flow at the port t_6 is $a > 0$, and valve v_1 is expected to be closed), level 5 is the *CLO*. The `Bottom-up` procedure considers the following nodes (in increasing level order): p_2, v_1, sc_1, sc_3, sc_5, sc_7, and sc_8. Diagnosing p_2 does not allow one to remove any of its behavioral modes, while diagnosing v_1 allows one to remove its *ok* and *stuckC* behavioral modes (*leak* is still possible, since we do not know anything about the output of v_1). Reasoning then proceeds to sc_1. Since its *ok* and *stuckC* modes abstract only situations in which its son v_1 is respectively *ok* and *stuckC*, then we can remove these modes also from sc_1 even if the only available observation is $s_{v_1} = closed$. Then, diagnosing sc_1 with the remaining modes does not allow to remove any of them. The levels used by *HD* and by the *BOTTOM-UP*

Table 3
Levels and corresponding number of candidates employed by HD and *BOTTOM-UP* in Example 12

Level	Number of candidates (HD)	Number of candidates (BOTTOM-UP)
5	6	5
4	16	8
3	64	32
2	256	64
1	1024	256
0	4096	1024

```
Procedure HIERARCHICAL-DIAGNOSIS
  Abstract-Observations;
  Rearrange;
  Bottom-up;
  Top-down-diagnosis.
```

Fig. 8. The *COMBINED* extension to Mozetic's algorithm.

extension with the associated number of components that have to be considered are shown in Table 3. The two algorithms exploit the same levels, but *BOTTOM-UP* is able to reduce the number of candidates that have to be considered at each level.

4.3.4. Combining the two strategies

The two proposed extensions to *HD* can be combined together in the following way: first, a tailored structural representation is built using the `Rearrange` procedure, and then the `Bottom-up` procedure is performed. The procedure that implements both strategies, which we call *COMBINED*, is shown in Fig. 8. By running this procedure on the previously analyzed diagnostic examples, one obtains all the advantages already described in the previous two sections.

4.4. Experimental evaluation

In this section, we describe the experimental activity that we carried out to evaluate the proposed algorithms. We compared *HD*, and its three extended versions *REARRANGE*, *BOTTOM-UP*, and *COMBINED*.

All algorithms were implemented in SWI Prolog [21] and run on a PowerPC G3 400 Mhz (Apple iMac DV) under the LinuxPPC 2000 operating system. The *Verify* procedure used by all algorithms was implemented as a generate-and-test algorithm, i.e., a procedure that considers each possible candidate and then verifies its consistency with the system description and the observations. Obviously, this is not an efficient implementation of the *Verify* function, but it has the advantage of giving us a worst case scenario (i.e., the exact size of the considered portion of the search space) for every diagnostic reasoning strategy we tested. The `Top-Down Diagnosis` and the `Abstract-Observations` procedures were the same for each algorithm.

The experimental comparison of the algorithms was performed with diagnosis problems of different sizes. We considered hydraulic systems, composed by volumetric pumps, pipes, valves, n-way nodes, sources, and sinks. The considered systems differed in the layout and in the number of components and ports. More precisely, the simplest system was the *hcs* (11 components and 13 ports), while the most complex system was a Heavy Fuel Oil Transfer System (HFOTS) (with 27 components and 25 ports) of a modern container ship. The layout and functioning of the HFOTS are described in [3].

For each hydraulic system, we considered several possible sets of observations. Each set was composed by a minimum of one observation and a maximum of N_{obs} observations, with

$$N_{obs} = trunc(0.4 * o)$$

where o is the number of ports for the considered system (e.g., in systems with 10 ports, each set was composed by a minimum of one and a maximum of four observations). The justification for this choice is twofold: first, with only a few observations, the problem is more difficult (and this is the typical case in which one needs to use abstraction); second, observations in real-world systems are often too few because sensors are too costly and/or too unreliable.

Each test was generated by first randomly choosing a number m of observations between one and N_{obs}, and then assigning a random value to m observables (randomly selected among the available ones). Each generated test was first checked in order to ensure that it had at least a solution, and then diagnosed using the four algorithms. For each algorithm execution, we recorded:

- total time spent (measured in seconds),
- total number of candidates considered for verification.

Fig. 9 illustrates the average time we obtained considering 1000 diagnostic scenarios for each considered hydraulic system. All three extensions perform better than *HD*: in particular, *REARRANGE* is significantly more efficient as the size of the considered system increases. This is due to the fact that as the number of components increases, *REARRANGE* has more possibilities of deriving additional levels for diagnosis. *BOTTOM-UP* obtains small improvements with respect to *HD*; as we hypothesized, the improvements obtained by *COMBINED* are better than both *REARRANGE* and *BOTTOM-UP*.

As one can see, for any algorithm the average time becomes very high for small increases in the number of components. This is due to the fact that we adopted a generate and test diagnostic strategy.

We now consider in more detail the experimentation on the most complex of the considered systems, i.e., the HFOTS. Our most detailed representation of the HFOTS was composed by one pump, 6 valves, 10 pipes, 1 three-way node, 2 four-way nodes, and 3 five-way nodes, with 2,985,984 possible candidates. Moreover, there were 43 ports and thus the number of observations ranged from 1 to 10. We built a hierarchy of structural

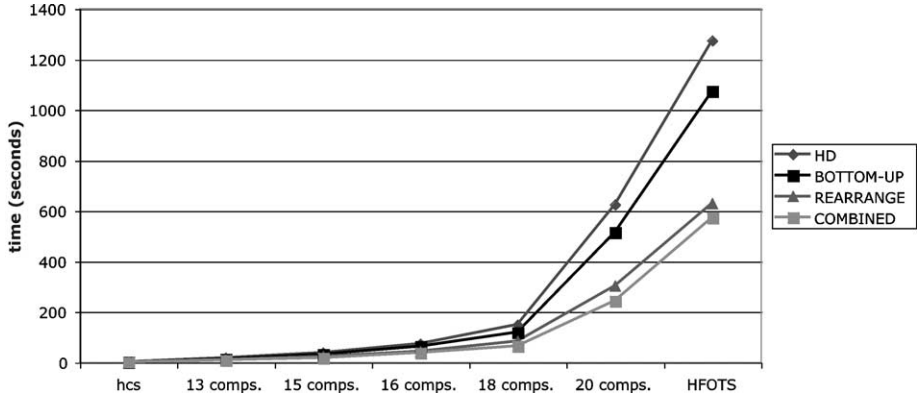

Fig. 9. Comparison using several hydraulic systems with different number of components.

Table 4
Number of candidates verified on the HFOTS considering 1000 diagnostic scenarios

Candidates	HD	REARRANGE	BOTTOM-UP	COMBINED
Average	238,878	117,529	172,372	89,234
Maximum	2,985,984	947,327	2,985,984	732,183
Minimum	240	240	180	180

Table 5
Time (in seconds) spent to diagnose the HFOTS considering 1000 diagnostic scenarios

Time	HD	REARRANGE	BOTTOM-UP	COMBINED
Average	1274	651	1073	496
Maximum	12742	3264	9376	2823
Minimum	2.13	2.15	1.74	1.77

abstractions for the HFOTS organized in 15 levels, where the most abstract level contains just one component representing the whole HFOTS.

Table 4 lists the average, maximum, and minimum number of candidates verified by each algorithm. Considering the average number of candidates, the value obtained with our extensions is 49% (*REARRANGE*), 72% (*BOTTOM-UP*), and 37% (*COMBINED*), of the value obtained with *HD*. The values of the maximum number of candidates can be explained as follows. In some cases, *HD* verifies all candidates at level 0: as we illustrated previously, this happens whenever it is not possible to abstract the available observations to upper levels. For the same reason, *BOTTOM-UP* exhibits a worst case which is identical to *HD*. The value for *REARRANGE* shows that the extension never reasons with just one level, i.e., it is able to exploit hierarchical reasoning in all considered scenarios. The value for *COMBINED* shows that, by combining the two extensions, one is able to further reduce the maximum number of verified candidates in the considered scenarios. The minimum values of candidates considered refer to situations where the hierarchy is fully exploited: thus, *HD* and *REARRANGE* obtain the same result. *BOTTOM-UP* (and thus *COMBINED*) is able in this case to slightly reduce the number of candidates that have to be verified.

Table 5 takes into consideration the time spent by each algorithm on the considered tests on the HFOTS. The average values show that all three extensions perform better than *HD* on the HFOTS. In particular, the values obtained by our extensions are respectively 52% (*REARRANGE*), 77% (*BOTTOM-UP*), and 39% (*COMBINED*) of the value obtained with *HD*.

Experimental evaluation on the HFOTS showed also that as the number of observations increases, the performance gains of *REARRANGE* over *HD* decrease, while those of *BOTTOM-UP* increase. This is explained as follows. As the number of observations increases, it is more likely that observations can be abstracted to coarser levels: in those cases, the performance of *REARRANGE* can only be slightly better than *HD*. On the other hand, it is also more likely that *BOTTOM-UP* is able to exclude more candidates with bottom-up reasoning, and thus to significantly improve performance over *HD*.

5. Conclusions

This paper considered structural abstraction in the context of model-based diagnosis, presented a formalization for it, and proposed an automated approach to the problem of obtaining effective structural abstractions, i.e., abstractions that are effective for reducing the computational effort which is required for the situation at hand. The effectiveness of the proposed algorithms was shown by experimental comparison with the most cited one in the literature, i.e., Mozetic's algorithm [14]. In the following, we outline some limitations of the experimental activity and directions for improvements.

First, the performed experimental evaluation needs to be extended by both considering other domains (such as electronics) and especially by considering other real-world systems. Moreover, the experimental evaluation needs also to be scaled up to systems with hundreds or thousands of components. In general, there is no reason to suspect that both the general advantages of hierarchical reasoning and those introduced by our extensions will not scale up considerably with the size of the system. Nevertheless, further experimental activity is needed to better evaluate the advantages of the proposed techniques in real-world situations.

Second, the experimental evaluation was performed by using a generate-and-test approach for the diagnosis of a single level, which caused generally high running times. Although this choice was taken to give us the exact size of the considered portion of the search space, it would be interesting to consider also more efficient approaches to diagnosis (such as the GDE approach). This would allow us to more precisely establish the performance gains that can be obtained in real-world diagnostic applications.

Finally, it must be noted that the tailoring of the hierarchical representation which is performed by our proposed algorithms may not always be the optimal one, and other criteria for tailoring the hierarchical representation could be tried. In its current formalization, the *REARRANGE* extension tries to derive more abstract levels without losing any available observations. Another possible criteria could be to allow losing observations in exchange for more abstract levels. The choice of which observations to keep in the abstract levels could be taken by considering the discriminating power of each observation, as described in [9].

References

[1] K. Autio, R. Reiter, Structural abstraction in model-based diagnosis, in: Proceedings of the Thirteenth European Conference on Artificial Intelligence (ECAI-98), Brighton, UK, Wiley, New York, 1998, pp. 269–273.

[2] L. Chittaro, R. Ranon, Hierarchical diagnosis guided by observations, in: Proceedings of IJCAI-2001, Seattle, WA, Morgan Kaufmann, San Mateo, CA, 2001, pp. 573–578.

[3] L. Chittaro, R. Ranon, A. Soldati, Introducing deviations and multiple abstraction levels in the functional diagnosis of fluid transfer systems, Artificial Intelligence in Engineering 12 (1998) 355–373.

[4] L. Console, P. Torasso, A spectrum of logical definitions of model-based diagnosis, in: W.H.L. Console, J. de Kleer (Eds.), Readings in Model-Based Diagnosis, Morgan Kaufmann, San Mateo, CA, 1992, pp. 78–87.

[5] R. Davis, Diagnostic reasoning based on structure and behavior, Artificial Intelligence 24 (1984) 347–410.

[6] R. Davis, W. Hamscher, Model-based reasoning: troubleshooting, in: W.H.L. Console, J. de Kleer (Eds.), Readings in Model-Based Diagnosis, Morgan Kaufmann, San Mateo, CA, 1992, pp. 3–24.

[7] J. de Kleer, Diagnosis with behavioral modes, in: Proceedings of IJCAI-89, Detroit, MI, Morgan Kaufmann, San Mateo, CA, 1989, pp. 104–109.
[8] J. de Kleer, A.K. Mackworth, R. Reiter, Characterizing diagnoses and systems, Artificial Intelligence 56 (1992) 197–222.
[9] J. de Kleer, B.C. Williams, Diagnosing multiple faults, Artificial Intelligence 32 (1987) 97–130.
[10] M. Genesereth, The use of design descriptions in automated diagnosis, Artificial Intelligence 24 (1984) 411–436.
[11] F. Giunchiglia, T. Walsh, A theory of abstraction, Artificial Intelligence 57 (2–3) (1992) 323–389.
[12] W.C. Hamscher, Modeling digital circuits for troubleshooting, Artificial Intelligence 51 (1991) 223–271.
[13] J. Mauss, M. Sachenbacher, Conflict-driven diagnosis using relational aggregations, in: Proceedings of the Tenth International Workshop on Principles of Diagnosis (DX-99), Loch Awe, Scotland, 1999, pp. 174–183.
[14] I. Mozetic, Hierarchical diagnosis, in: W.H.L. Console, J. de Kleer (Eds.), Readings in Model-Based Diagnosis, Morgan Kaufmann, San Mateo, CA, 1992, pp. 354–372.
[15] P.P. Nayak, A.Y. Levy, A semantic theory of abstractions, in: Proceedings of IJCAI-95, Montreal, QB, Morgan Kaufmann, San Mateo, CA, 1995, pp. 196–203.
[16] C.J. Price, N.S. Taylor, Multiple fault diagnosis using fmea, in: Proceedings of AAAI-97, Providence, RI, AAAI Press, 1997, pp. 1052–1057.
[17] R. Ranon, The closure properties of functional flow-based approaches and their relevance to diagnosis, in: Proceedings of the Thirteenth European Conference on Artificial Intelligence (ECAI-98), Brighton, UK, Wiley, New York, 1998, pp. 289–290.
[18] R. Reiter, A theory of diagnosis from first principles, in: W.H.L. Console, J. de Kleer (Eds.), Readings in Model-Based Diagnosis, Morgan Kaufmann, San Mateo, CA, 1992, pp. 29–48.
[19] P. Struss, What's in sd? towards a theory of diagnosis, in: W.H.L. Console, J. de Kleer (Eds.), Readings in Model-Based Diagnosis, Morgan Kaufmann, San Mateo, CA, 1992, pp. 419–449.
[20] M. Stumptner, F. Wotawa, Guest-editorial special issue on industrial applications of model-based reasoning, AI Comm. 13 (2).
[21] Swi prolog, University of Amsterdam, http://www.swi-prolog.org.
[22] L. Travé-Massuyés, T. Escobet, R. Milne, Model-based diagnosability and sensor placement application to a frame 6 gas turbine subsystem, in: Proceedings of IJCAI-2001, Seattle, WA, Morgan Kaufmann, San Mateo, CA, 2001, pp. 551–556.