AGH
University of Science and Technology

Faculty of Electrical Engineering, Automatics,
Computer Science and Electronics

Jakub Oleksiak

Ph.D. Thesis

Hierarchical diagnosis of technical systems on the basis of model and expert knowledge

Supervisor:
Prof. Antoni Ligęza, Ph.D.

Kraków, 2004

,

Contents

Part I

Introduction

Chapter 1

Introduction

The history of diagnosis is probably as old as human history. The term "diagnosis" can be defined in the simplest way as an *examination of systems in order to perform detection and localization of their malfunctions*. At first, there was only one object of examinations: human itself. Diagnostic procedures were reduced to use expert knowledge of a medicine-man or a quack. But the domain of diagnosis enlarged, together with evolution of the technical civilization, and started including technical systems and technological installations.

Technical systems are exposed to many external and internal factors which are causes of slow deterioration of products and possible malfunctions of systems. We can assume that all malfunctions of technical systems, which occur during a normal work, are undesirable and harmful. The malfunctions either reduce quality of produced goods or, in the worst case, definitely interrupt the production process what generates measurable economic losses. The value of loses is sometimes so huge that growing interest in effective diagnostic methodologies is understandable.

The most serious difficulties with developing good diagnostic methodology for technical systems are: growing complexity of their structures and variety of technologies in use. These two factors make a diagnostic process very hard, even for well trained specialists, and increase time which is necessary for diagnosis and reparation, as well as complexity of the diagnostic procedures.

In face of the above facts, developing efficient diagnostic methodologies becomes a very important task. Efficient diagnostic methodology should be able to support human diagnostic actions or even carry out diagnosis quite automatically. A growing number of researchers work in this area, which has appeared as an interdisciplinary branch of science and engineering during the last three decades. The increasing number of research centers involved in this area of research is a good proof that diagnosis is one of the most important problems of modern science.

A typical diagnostic procedure can be divided into the following three principal stages (Isermann and Ballé, 1996):

- *fault detection* - detection of system misbehaviors,
- *fault isolation* - localization of detected system misbehaviors,

- *fault identification* - estimation of size and type of localized misbehavior.

In technical literature referring to diagnosis, fault detection and isolation is often referred to as *FDI* activity.

Generally speaking, the idea of *fault detection* is based on comparison of current behavior of a system with the expected behavior of the system. If there are observable differences, then one can assume that the system does not work correctly. In other words, the *observed behavior* of the system is inconsistent with the *expected* one.

The next step is *fault isolation*; parts of the system which are responsible for the observed misbehaviors are localized. The misbehavior is precisely described during *fault identification*. *Fault isolation* and *fault identification* are sometimes joined together and called *fault diagnosis*.

Existing diagnostic methodologies can be divided according to a few different criteria. The most general ones are: the type of diagnostic knowledge and the type of searching strategy. Below, the diagnostic methodologies are grouped according to the type of diagnostic knowledge which describes system behavior or relations between symptoms and reasons of defects:

1. Systems described by a well developed model. This class of systems can be diagnosed most precisely and in a complete way, but only very limited number of systems have model which is good enough for such diagnosis. The most typical solutions applied in practice are usually based on:
   - analytical models, such as:
     - physical equations,
     - linear input–output models (transfer function),
     - linear state equations e.g. (Chow and Willsky, 1984),
     - state observers and Kalman filters e.g. (Chen and Patton, 1999).
   - knowledge–based models e.g. (Reiter, 1987; de Kleer and Williams, 1987; Genesereth, 1993).
2. Systems described only by general rules. The knowledge is derived from domain experts. Applied solutions are based mainly on knowledge engineering and computational intelligence:
   - casual graphs e.g. (Ligęza and Fuster-Parra, 1997),
   - fuzzy logic e.g. (Kościelny *et al.*, 1999),
   - diagnostic matrices e.g. (Kościelny, 2001),
   - ATMS e.g. (de Kleer, 1986),
   - diagnostic trees and graphs e.g. (Barlow and Lambert, 1975),
   - classical AI rule–based expert systems e.g. (Tzafestas Ed., 1989; Liebowitz, 1998).
3. Systems that are not described by any model or rules; all our knowledge of the systems is based only on observations of their behaviors. Applied solutions:

- pattern recognition e.g. (Korbicz, 1998),
- neural networks e.g. (Sorsa and Koivo, 1993),
- fuzzy neural networks e.g. (Kościelny, 2001).

The most important for us are approaches derived from logical reasoning methods. Especially, *knowledge–based models* and *causal methodologies* are further used in this thesis. This is so, since precise mathematical models are available only for limited number of systems, while knowledge engineering and computational intelligence methods are, in principle, applicable to any system of arbitrary complexity (although the level of precision may vary depending on the complexity of the system).

The knowledge-based models for diagnosis use inconsistencies between observed and expected behavior of a system where expected behavior is predicted according to knowledge of a system model. This kind of approaches is also called *consistency–based diagnosis*. Such diagnostic systems do not need the expert knowledge acquisition phase or training and can diagnose even very new systems for which expert diagnostic knowledge, based on former experience, does not exist. For such systems, existing models of their correct behavior, developed during design and simulation of such systems, can be used for diagnosis.

The main idea of presented further diagnostic models is based on components, similar to (Reiter, 1987), where system is defined as a set of components and relations among them. Diagnosis of such a component system is done in two stages: *fault detection* and *fault isolation*. The model of a system has some selected inputs and outputs. Values of system inputs are measured and used as input for the model. Values of model outputs are calculated and compared with observed system outputs. If there are significant discrepancies, then system malfunction is detected. A general idea of such a model-based approach is presented in Figure 1.1.

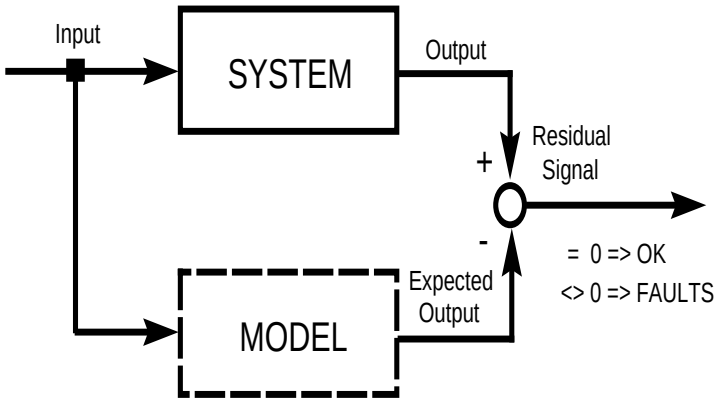

Figure 1.1: General idea of consistency–based diagnosis

In case of detection of faulty behavior, *fault isolation* is to be performed. Sets of components which cannot work correctly are identified (the so-called *conflict sets* (Reiter, 1987)). *Conflict sets* include components participating in generating outputs of the system

at which discrepancies were detected. The elements of conflicts sets are natural candidates for being faulty (at least one of them), and a potential diagnosis has to include at least one component from each conflict set.

On the other hand, *Causal methodologies* model cause–effect relations which describe correct and incorrect behavior of the systems under examination. The basic logic functors (AND, OR, NOT) are usually used for modeling relations between symptoms and reasons of faults. Simplicity of this approach is its biggest advantage but sometimes this formalism is too simply for describing larger systems.

The expert logical graphs constitute an example of the causal methodology. They constitute graphical representations of causal relations (Fuster-Parra, 1996; Ligęza, 2003). An expert graph is defined as an acyclic, directed graph with nodes modeling symptoms and arcs referring to causal relations. The types of nodes in such graphs can be: AND (conjunction), OR (disjunction), and NOT (negation). AND nodes represent logical conjunctive influence of predecessor nodes on the modeled symptom, OR nodes represent disjunctive influence, while NOT arcs represent negative (reversed) influence. Diagnoses are generated by use of simply rules for propagation of node states through an expert graph combined with search for logical values of input nodes implying the observed misbehavior. The AND node is represented in figures as a node with additional arc under the node, the OR is a pure graph node, and the NOT arc is represented by a arc with black dot.

1.1 Problem description

Diagnosing of modern technological systems becomes more and more difficult and sophisticated task. This is so mainly due to their complexity – contemporary technological systems are assembled from numerous components which cooperate and recursively include other components.

Diagnostic descriptions based on the mentioned in previous section diagnostic methodologies are usually flat. This means that all components are analyzed simultaneously as a potential source of faults. When the number of components increases then both complexity of diagnostic reasoning and the time of calculation also increases regardless of necessary precision of diagnoses and potential influence of new components on the observed symptoms. Single–level diagnostic procedure generates many potential diagnoses which cannot be verified in a short time.

The problem increasing complexity can be solved by hierarchization of model where the position in vertical hierarchy represents some level of considered details. Hierarchical approaches for solving diagnostic problems were considered by few authors only, e.g.: theoretical description of hierarchical approaches in (Giunchiglia and Walsh, 1989), hierarchical approach based on subsumption between first–order clauses in (Plaisted, 1980) or hierarchical diagnosis based on constraints in (Mozetič, 1991), but none of these approaches proved to be applicable in practice. A recent approach is also presented in (Kościelny, 2004) in the context of FDI based on use of expert-defined *diagnostic matrix* and its decomposition to subsystem matrices.

In this thesis another, model-based hierarchical approach is considered. It is based

on direct modeling of the hierarchy of system components which can recursively include other components (e.g. Figure 1.2). The diagnostic procedure can refer to a certain *level* of component representation. The system to be diagnosed is structured in a *hierarchy of components* with respect to its structure and functionality.

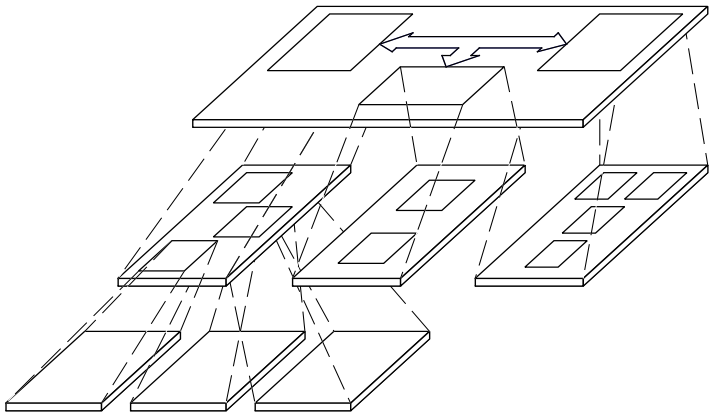

Figure 1.2: Abstract example of hierarchical model

Note that some components of a complex system may have a good diagnostic description and some of them may have only a partial one or even none, e.g. some components come from different manufacturers or represent quite a different way of operation than others. If we have a look at the presented before classification then we can see that there are approaches emerging from classical theory of control and also many approaches belonging to "soft computing" class. Hence, it is necessary to find a way of simultaneous exploration of different diagnostic models for generating diagnoses for so different components in the same hierarchical model.

1.2 The aims and scope of the thesis

The thesis is concerned with diagnosis of technical systems with use of *Artificial Intelligence* approaches. The main focus is on hierarchical diagnosis allowing for multi-level analysis.

The main goal of this dissertation is to provide a deeper theoretical insight into hierarchical modeling and diagnostic reasoning. This goal is to be accomplished in the following stages:

- detailed analysis of the state of the art and existing AI approaches to diagnosis, especially consistency-based ones and causal ones,
- presenting definition and a formal concept of a new generic model for modeling hierarchical structures of complex systems to be used in diagnostic inference,
- developing a methodology for hierarchical diagnostic reasoning,

- developing conceptual experimental software supporting hierarchical modeling and analysis of complex technical systems.

Moreover, auxiliary study of computational complexity, probabilistic aspects, testing and verification of diagnostic reasoning in case of hierarchical approach is presented.

With reference to the above stated goal, and on the basis of thorough analysis of the state of the art of AI based diagnostic methodologies, the thesis of the dissertation is formulated as follows:

In case of complex technical systems, it is rational to apply hierarchical diagnostic approach in order to reduce complexity. An efficient hierarchical diagnostic methodology can be developed through composition of consistency-based methods and causal AND/OR/NOT graphs into hierarchical multi-level diagnostic procedure. Such an approach allows for qualitatively new, flexible analysis of complex systems failures and efficient in-depth diagnosis.

Hierarchical multi-level diagnostic procedure combines graph-search methodologies (for traversing different levels of details with advanced AI-based diagnostic inference methodologies). During the search for diagnoses it is possible to focus on selected components and penetrate the internal structure of them, or stop the procedure at some required levels of details.

The specific advantages of such an approach include the following innovative features:

- diagnostic reasoning is split into levels; at a specific level the most appropriate specific diagnostic approach can be selected and applied,
- thanks to reduction of complexity, diagnostic efficiency is improved since at a specific level only limited number of components (identified by higher level diagnosis) is to be considered,
- the degree of details of the specific diagnosis can be adjusted to current requirements and available diagnostic observations,
- hierarchical approach provides significant improvement of scalebility in case of complex systems,
- current analysis can be focused on subsystems (subcomponents) while other ones are not taken into consideration,
- the proposed hierarchical methodology allows to incorporate auxiliary mechanisms for improving efficiency; for example control of diagnostic reasoning based on probabilistic information or verification procedures.

Last but not least, hierarchical analysis allows for more efficient knowledge representation and processing with use of computer applications. This concerns, among other issues, screen based communication through graphical interface of limited size[1].

[1]Even biggest standard 21" computer displays allow to present only limited part of large models of industrial systems, some of them having several thousands of components

The Reiter's classical definition of component system introduced by Reiter (1987) puts forwards a *single-level* view on system modeling – the system is composed of a number of equally-ranked, atomic components, connected and interrelated. The behavior of the system is modeled with a *single-level,* flat first-order theory. No internal structure of components is considered.

For the sake of *hierarchical diagnosis* it is proposed to extend Reiter's definition of *system* over *complex systems,* recursively composed of components having some internal structure. A complex system *CS* is represented here by a set of top–level, interrelated components.

In the classical Reiter approach elements of the set of *COMPONENTS* are considered to be of atomic nature; here *complex components* and *elementary components* will be introduced.

Complex components can be described here by two kinds of logic–based knowledge representations:

- Model–based ones with Reiter's theory as a diagnostic procedure (Reiter, 1987),
- Expert–based causal logical graphs with propagation of values in AND/OR/NOT graphs as a diagnostic procedure (Fuster-Parra, 1996)).

The model-based description does not require causal modeling, but it is computationally harder and not all technological components have an appropriate model useful in this kind of approach. The expert causal graphs give possibility to create an efficient diagnostic description based on human causal knowledge about relations between reasons of faults and their observed manifestations. The most important disadvantage is also lack of graphs for some components, and difficulties with building description for more complex systems with functional dependencies among components. In this thesis both of the approaches are combined to enable hierarchical diagnostic procedure.

Original results of the thesis include:

1. Development of a formal framework and an original methodology for hierarchical modeling of complex systems with heterogeneous diagnostic descriptions,
2. Design of an efficient diagnostic procedure with "focus" effect (diagnosis of subsystems as separate systems) for the hierarchical model,
3. Development of a verification procedure for the hierarchical diagnostic methodology. The procedure uses properties of the hierarchical model,
4. Development of a methodology for using expert knowledge and defect statistics for ordering diagnoses and improving efficiency of the diagnostic procedure,
5. Design of a computer application for supporting diagnostic procedure with hierarchical modeling of complex systems.

Moreover, a number of detailed concepts and definitions is introduced. A critical review of existing, flat methodologies is provided.

1.3 The outline of this thesis

This thesis is structured into four main parts, the bibliography and appendices:

1. Part I: Introduction.
2. Part II: Survey of diagnostic methodologies.
3. Part III: Hierarchical diagnosis.
4. Part IV: Examples and conclusions.

The first part is just one chapter of introduction to this work. A rough survey of diagnostic methodologies and general problems in diagnostic domain are presented. The chapter has as its main objective to present an outline of the contents of this thesis.

The second part is composed of three chapters. In this part a critical revision of different works of diagnosis is presented.

- Chapter 2. Methodologies of diagnosis - general short survey of selected diagnostic methodologies. It provides a list of most important logic–based diagnostic methodologies and presents in brief basic ideas of every approach.
- Chapter 3. Causal AND/OR/NOT graphs - more detailed discussion about diagnostic methodologies based on classical AI – casual AND/OR/NOT graphs. The graphs are one of two proposed descriptions of complex components.
- Chapter 4. Model–based approaches - description of two well known model–based diagnostic approaches. The main idea of this thesis is close to model–based component diagnosis. Moreover, model–based description is one from two proposed alternative descriptions of complex components.

The third part is the main part of this thesis and it is composed of four chapters, among them the most important one is Chapter 5. This part has as its main objective presentation of a model for hierarchical diagnostic reasoning which constitutes the main proposal of this thesis.

- Chapter 5. Hierarchical diagnosis in heterogenic environment - introduces the most important ideas and formal background for the proposed approach to diagnostic knowledge representation. The main objective of this chapter is presentation of the formalism for hierarchical modeling of complex systems.
- Chapter 6. Inter–level mapping for consistency reasoning and casual graphs. A short presentation of the form of mapping functions for two types of diagnostic description: models and causal graphs. The mapping functions pass diagnostic information between levels of the hierarchical model.
- Chapter 7. Diagnostic algorithm - presents a way of diagnostic inference for hierarchical models. An adequate diagnostic algorithm is introduced.

- Chapter 8. Diagnostic process - includes some other elements of hierarchical diagnosis such as: fault detection and isolation, verification of diagnoses, methodology of tests, and ordering diagnoses according to probabilistic information.

The fourth part is oriented to examples, some conclusions and further work. Proposal of a diagnostic application is also presented there. It is composed of two chapters:

- Chapter 9. Practical examples - simplified examples illustrate how this theory can be used to establish diagnoses. The computer application for modeling complex systems is also presented.
- Chapter 10. Concluding remarks and further work - A summary of original results of the thesis is provided and future lines of work are established.

The references used in this work have been presented in the chapter of Bibliography. Finally, appendices are presented:

- Appendix A - Reiter's methodology to calculation of diagnoses,
- Appendix B - the appendix describes some theoretical approaches to hierarchization.

Part II

Survey of diagnostic methodologies

Chapter 2

Methodologies of diagnosis

Diagnostic methodologies emerge from almost all domains of modern Computer Science and Control Theory. Some of them are mentioned in Chapter 1, but the number of original approaches and hybrid approaches is enormous and might be a subject of a separate book. For example, an interesting survey of diagnostic methods is presented in (Venkatasubramanian *et al.*, 2003); a survey of terminology and classifications of diagnostic approaches is shown in (Kościelny and Szczepaniak, 1997).

This chapter presents just a few, selected from many others, approaches to diagnosis. The selection criteria include mainly type of diagnostic procedures and how diagnostic inference is close to methods used in classical logic. Presented approaches are mainly *fault isolation* approaches.

2.1 Diagnostic matrices

Binary diagnostic matrices (Gertler, 1998; Kościelny, 2001) constitute tools belonging *fault isolation* group of activities. In this approach, causes of a fault are determined on the basis of residual signals from a *fault detection* algorithm, and knowledge about their relations to real causes of faults.

Let us introduce this approach in more details. Let F denote a set of possible faults of a system,

$$F = \{f_k : k = 1, 2, \ldots, K\},$$

and let S be a set of diagnostic (residual) signals that are outputs of detection algorithms,

$$S = \{s_j : j = 1, 2, \ldots, J\}.$$

The diagnostic relation R_{FS}, being the core concept in the presented approach, is defined on the Cartesian product of the sets F and S, i.e.

$$R_{FS} \subset F \times S.$$

The relation has a matrix form; the matrix element $r(f_k, s_j)$ is defined as follows:

$$r(f_k, s_j) \equiv v_j(f_k) = \begin{cases} 0 \Leftrightarrow \langle f_k, s_j \rangle \notin R_{FS}, \\ 1 \Leftrightarrow \langle f_k, s_j \rangle \in R_{FS}. \end{cases}$$

Table 2.1: A binary diagnostic matrix for the three–tanks system (Kościelny, 2001)

S/F	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
s_1	1				1	1	1	1						
s_2	1	1	1						1			1		
s_3		1	1	1					1	1			1	
s_4			1	1						1	1			1

Table 2.2: Example of FIS system (Kościelny, 2003)

S/F	f_1	f_2	f_3	f_4	f_5	f_6	f_7
s_1	1	0	1	0	0	1	{0, 1}
s_2	0	-1	0	+1	-1	0	{0, +1, -1}
s_3	-1	+1	+1, -1	0	+1	+1	{0, +1, -1}
s_4	0	1, 2	0, 1	0	1, 2	1, 2	{ 0, 1, 2}
s_3	+1	0	+1	+1	0	+1, -1	{0, +1, -1}

We shall also say that the set of diagnoses is constituted as follows:

$$DGN = \{f_k \in F : \forall_{j:s_j \in S}[v_j(f_k) = v_j\}$$

Diagnostic matrices can be determined on the basis of residual equations or expert knowledge. A binary diagnostic matrix for example system (the classical three tanks system, see Kościelny (2001)) is presented in Table 2.1.

The basic approach with 0 and 1 values can be extended towards incorporating a more detailed characteristics of diagnostic signals. In this case the values of diagnostic signals can take more than two values (e.g. $-1, 0, +1$). Then it is possible to carry out diagnostic process based on *Fault Information Systems* FIS. The *Fault Information System* is based on *Information Systems* IS (Pawlak, 1983) and defined as a four-tuple:

$$FIS = \langle F, S, V_S, r\rangle,$$

where $V_S = \bigcup_{j:s_j \in S} V_j$ is a set of values of all diagnostic signals, V_j is the domain of diagnostic signal s_j, and r is a function such that

$$r : F \times S \to V_S,$$

and

$$\forall_{f \in F, s \in S}\, r(f, s) \in V_S.$$

Each pair $\langle f, s\rangle$ has assigned a single value or a set of values. In the first case, such system is called *simple* while in the second – *rough*. An example diagnostic matrix for a rough system is presented in Table 2.2.

The main disadvantage of diagnostic matrices is that their residual signals are very specific for the type of the system to be diagnosed and matrices have to be modified when some parts of the system are changed. Moreover, fault localization for this methodology is mainly based on expert knowledge and can be ambiguous or incomplete for some cases.

2.2 Fault trees

The idea concerning this approach is taken from (Barlow and Lambert, 1975). Fault tree analysis, FTA, evolved in the aerospace industry in the early '60's; at present, it is one of the principal methods of systems safety analysis. It can predict the most likely causes of system failure in the event of a system breakdown. The goal of fault tree construction is to model the system conditions that can result in an undesired event.

A fault tree is a model that graphically and logically represents various combinations of possible events, both fault and normal, occurring in a system that lead to the top event. An example fault tree is presented in Figure 2.1.

The term event, denotes a dynamic change of state that occurs to a system element. A fault event is an abnormal system state. A normal event is an event that is expected to occur. Different kinds of event symbols are used:

- A rectangle defines an event that is the output of a logic gate and is dependent on the type of logic gate and the inputs to the logic gate.
- A circle defines a basic inherent failure of a system element when operated within its design specifications.
- A diamond represents a failure, other than a primary failure that is purposely not developed further.
- A switch event represents an event that is expected to occur or to never occur because of design and normal conditions, such as a phase change in a system.

The fundamental logic gates for fault tree construction are the OR and the AND gates. The OR gate describes a situation where the output event will exist if one or more of the input events exist. The AND gate describes the logical operation that requires the coexistence of all input events to produce the output event.

2.3 ATMS

TMS (Truth Maintenance System) (de Kleer, 1986) and ATMS (Assumption-based Truth Maintenance System) (de Kleer, 1986) have been used in problem solving. Several authors have used ATMS in their diagnostic systems, among them: (de Kleer and Williams, 1987), (Struss, 1988), etc. TMSs are devoted to find one solution, ATMSs are devoted to find all the solutions.

ATMS context is the set formed by the assumptions of a consistent environment combined with all nodes derivable from those assumptions. A characterizing environment for a context is a set of assumptions from which node of the context can be derived. An ATMS label is a set of environments associated with every node.

The basic data structure is an ATMS node. It contains the problem solver datum with which it is associated, the justifications the problem solver has created for it, and a label computed for it by the ATMS. All nodes are treated identically, and are distinguished only by their individual pattern of label environments and justifications.

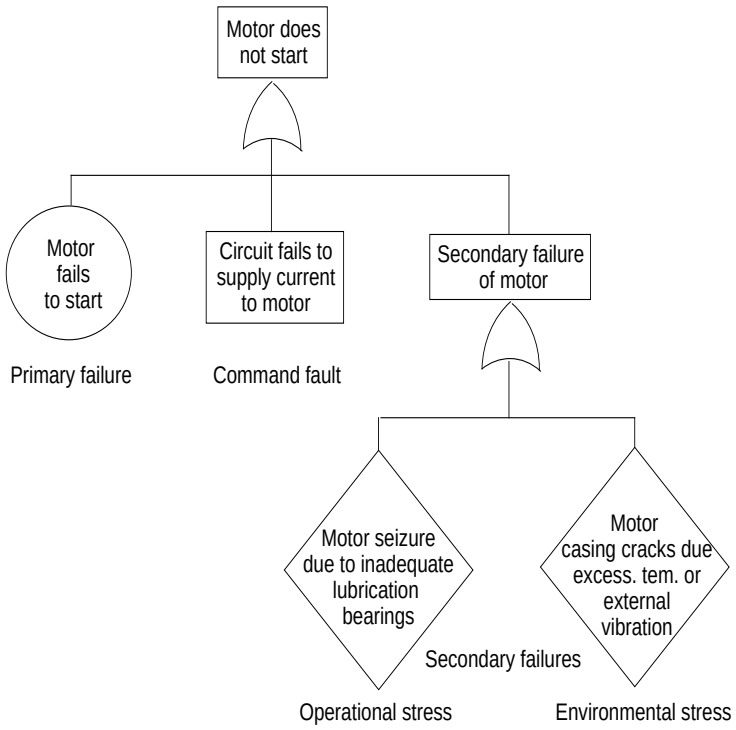

Figure 2.1: Example of a fault tree

There are four types of nodes: premises, assumptions, assumed nodes, and derived nodes. The ATMS associates every datum with its contexts. If a datum is in a context, then it is in every superset as well (the inconsistent supersets are ignored). The three basic ATMS actions are creating an ATMS node for a problem–solver datum, creating an assumption, and adding a justification to a node.

Qualitative reasoning and constraint languages, both involve choosing among alternatives. LOCAL (de Kleer, 1976) is a program for troubleshooting electronic circuits which was incorporated in SOPHIE III (Brown, 1982). It uses propagation of constraints to make predictions about device behavior from component models and circuit measurements. When a fault of the circuit occurs, some of its components is not operating as intended. Thus, at some point, as the correct component model does not describe the actual faulty component, the predictions will become inconsistent. The assumptions are that individual components are functioning correctly. A contradiction implies that some assumption is violated. Hence the fault is localized to a particular component set. The best measurement to make next is the one that provides maximal information of the yet unverified assumptions. This program requires that the assumptions enabling diagnostic inference have to be explicitly available.

2.4 Expert systems

Expert systems are usually rule–based systems with large knowledge base including *if–then* rules. Expert diagnostic knowledge is codified in these rules associating symptoms with underlying faults. Further diagnosis is done by either forward or backward reasoning based on observations and the rules.

The main stages in an expert system development include:

- knowledge acquisition,
- choice of knowledge representation,
- coding of knowledge in the knowledge base.

This approach, similar to *fault trees* and *ATMS*, needs knowledge acquisition stage and strong validation of acquired knowledge (incompleteness, inconsistency). Moreover, the number of rules grows rapidly with complexity of the system to be analyzed. Such diagnostic systems fails when new, not defined in knowledge base, condition is encountered, i.e. knowledge collected in rules describes rather experience concerning a specific device than general knowledge of structure or behavior.

Methodologies of validation and design of rule–based systems with some examples are described in e.g. (Ligęza, 1996a,b). Designation and verification of rule–based systems can be supported by specialized software. There are many approaches to graphical representation of rules and dependencies among of them e.g. (Oleksiak and Ligęza, 2001).

2.5 Case–Based Reasoning

The basic idea of case–based reasoning is that if two problems are similar then it is possible to solve the first problem by using adapted solution of the second problem (Kolodner, 1983). Hence, it is a form of reasoning by analogy. The set of previously solved cases constitutes the knowledge base. When a specific problem occurs, a similar situation in the past is searched for in the knowledge base. If such approximately similar situation is found then its past solution is adapted for the current problem situation.

The case–based reasoning is successful used in diagnosis, examples can be found in (Bach and Allemang, 1996).

This approach needs a case acquisition phase and the form of knowledge can vary depending on the diagnosed system.

2.6 Constraints and constraint abstraction

Mozetič proposed hierarchical diagnosis based on constraints where abstraction is reached by collapsing of values, deletion of variables and simplification of levels (Mozetič, 1991). A little bit different component hierarchy can be found also in (Mozetič, 1989).

Abstraction of system is built there on the basis of constraints m between independent variables x (system states) and depend variables y (input–output observations). More abstract levels have more general constraints. Operator h is a hierarchy operator which maps variables between two levels. An idea of two level model is shown in Figure 2.2. There are abstract level 1 and detailed level 2.

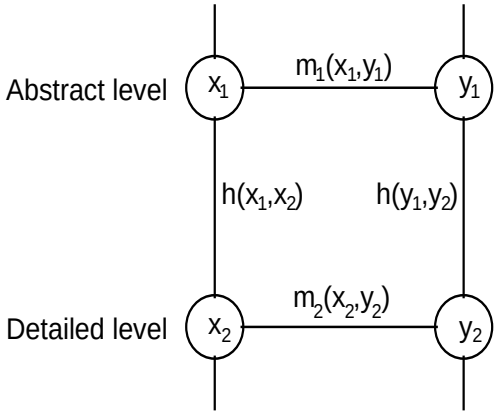

Figure 2.2: Relations of abstraction (Mozetič, 1989)

Example 2.1 *(Mozetič, 1989) The model of an OR gate has an abstract level and a detailed level. The abstract level includes logical disjunction of two inputs. Values of inputs and outputs are either logic "0" or logic "1". Constraints in this level are:*

$org1(1,1,1).$
$org1(1,0,1).$
$org1(0,1,1).$
$org1(0,0,0).$

The detailed level is an electric circuit with some transistors and some passive elements. Using a Prolog-like notation the model of the circuit can be specified as follows:

$org2(vi(V_{in1},I_{in1}),vi(V_{in2},I_{in2}),vi(V_{out},I_{out})) \leftarrow$
$V_{cc}=5, V_e=0,$
$resistor(V_{in1},V_{b1},I_{in1},4700),$
$transistor(V_{b1},V_c,V_e,I_{in1},I_{c1},I_{e1}),$
$resistor(V_{in2},V_{b2},I_{in2},4700),$
$transistor(V_{b2},V_c,V_e,I_{in2},I_{c2},I_{e2}),$
$resistor(V_{cc},V_c,I_{cc1},470),$
$I_{cc1}=I_{c1}+I_{c2}+I_{b3},$
$resistor(V_c,V_{b3},I_{b3},4700),$
$transistor(V_{b3},V_{out},V_e,I_{b3},I_{c3},I_{e3}),$
$resistor(V_{cc},V_{out},I_{cc2},470),$
$I_{cc2}=I_{c3}+I_{out}, \quad 0 \leq I_{out},I_{out} \leq 0.006.$

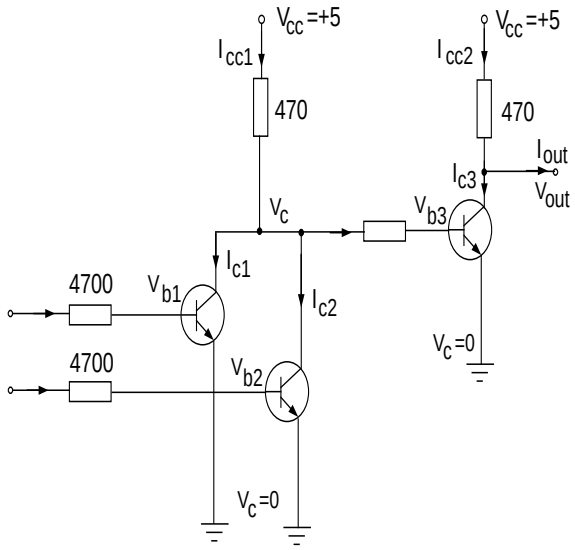

Figure 2.3: The OR gate realized by three npn transistors

$transistor(V_b, V_c, V_e, I_b, I_c, I_e) \leftarrow$ %*active*
$V_b = V_e + 0.7, \%V_{be} = 0.7$
$V_c \geq V_b,$
$I_c = 100 \times I_b, \%Beta = 100$
$I_b \geq 0,$
$I_e = I_c + I_b.$

$transistor(V_b, V_c, V_e, I_b, I_c, I_e) \leftarrow$ %*saturated*
$V_b = V_e + 0.7, \%V_{be} = 0.7$
$V_c = V_e + 0.3, \%V_{cesat} = 0.3$
$I_b \geq 0,$
$I_c \geq 0,$
$I_e = I_c + I_b.$

$transistor(V_b, V_c, V_e, I_b, I_c, I_e) \leftarrow$ %*cutoff*
$V_b < V_e + 0.7, \%V_{be} = 0.7$
$I_b = 0,$
$I_c = 0,$
$I_e = 0.$

$resistor(V_1, V_2, I, R) \leftarrow$
$R > 0,$
$V_1 - V_2 = I \times R.$

Hierarchical relations between variables in individual levels are specified by the following two clauses:

$h(0, vi(V, I)) \leftarrow 0 \leq V, V < 0.7.$
$h(1, vi(V, I)) \leftarrow 2 \leq V, V \leq 5.$

A diagnostic process can be carried out by using the below clause:

$org1(In1, In2, Out) \leftarrow$
$h(In1, VI_{in1}),$
$h(In2, VI_{in2}),$
$h(Out, VI_{out}),$
$org2(VI_{in1}, VI_{in2}, VI_{out}).$

Mozetič puts two consistency conditions on his abstraction:

$$C1: \ \forall x_2, y_2 \ \ m_2(x_2, y_2) \ \Rightarrow \ \neg\exists x_1 \ h(x_1, x_2) \vee \exists y_1 \ h(y_1, y_2),$$
$$C2: \ \forall x_2, y_2 \ (\exists x_1', y_1' \ m_2(x_2, y_2) \ \wedge \ h(x_1', x_2) \ \wedge \ h(y_1', y_2)) \Rightarrow$$
$$\exists x_1, y_1 \ m_1(x_1, y_1) \ \wedge \ h(x_1, x_2) \ \wedge \ h(y_1, y_2).$$

The $C1$ condition restricts incompleteness introduced by the hierarchical operators and prohibits cases where detailed independent variable x_2 with abstraction x_1 is mapped to detailed dependent variable y_2 without abstraction. The condition is necessary because the abstract level does not include some variables or values of variables which are in detailed level. Therefore, the y_1 has to exist when x_1 exists for keeping consistency.

Example 2.2 *Values of inputs and outputs in the abstract model are either logic "0" (0V, 0.7V) or logic "1" (2V, 5V). Any electrical voltage can appear on the detailed model inputs, but output will be either (0V, 0.3V) or (2.18V, 5V) what corresponds to the two logic states. Therefore any y_2 has an abstraction and the condition $C1$ is satisfied for this model.*

The $C2$ condition guarantees that if there exists abstraction of independent variable x_2 and dependent variable y_2 and relation m_2 among them, then there have to exist also an abstract independent variable x_1 and an abstract dependent variable y_1 and relation among them m_1, such that the independent variable for x_1 is x_2 and the dependent variable for y_1 is y_2.

Example 2.3 *All detailed values of all variables having constraints defined at the detailed level, and which have abstraction, must also have constraints at the abstract level e.g.:*

- *voltages in the detailed level are:*
  $V_{in1} = 0.5$, $V_{in2} = 3.0$, $V_{out} = 3.0$,
- *logic values in the abstract level are:*
  $In1 = 0$, $In2 = 1$, $Out = 1$.

There exist constraints at the abstract level for the above values, and it is of the form $org1(0,1,1)$.

This type of abstraction is used in hierarchical version of the *CARDIO* heart model (Bratko *et al.*, 1989) which is qualitative model of electrical activity of the heart. The model maps any arrythmia to all corresponding ECG descriptions.

Limitation of abstraction based on constraints is that modeling of systems is not easy and expert knowledge must be acquired and codified.

2.7 Hierarchical diagnostic matrices

Hierarchical extension of diagnostic matrices is described in (Kościelny, 2004). A diagnosed system is divided into h levels. Let M_h denote the number of subsystems at the h-th level. A subsystem n at level h is described by the following three-tuple:

$$O_n^h = \langle F_n^h, S_n^h, R_{FS}^{h/n} \rangle$$

where:
F_n^h is a subset of faults for subsystem n at level h,
S_n^h is a subset of diagnostic signals that they detect faults for subsystem n at level h,
$R_{FS}^{h/n}$ is their diagnostic relation.

The above hierarchical structure has the following properties:

- subsets of diagnostic signals are separate in all subsystems:

$$S_m^g \cap S_n^h = \emptyset,\ m,n = 1,\ldots,M_h,\ m \neq n,\ g,h = 1,\ldots,H$$

- faults–diagnostic signals relations are also separate:

$$R_{FS}^{g/m} \cap R_{FS}^{h/n} = \emptyset,\ m,n = 1,\ldots,M_h,\ m \neq n,\ g,h = 1,\ldots,H$$

- subsets of faults detected in subsystems at the same hierarchical level are separate:

$$F_m^h \cap S_n^h = \emptyset,\ m,n = 1,\ldots,M_h,\ m \neq n,\ h = 1,\ldots,H$$

The hierarchical diagnosing structure is described by the graph:

$$GHSD = \langle O, L \rangle,$$

Subsystems are represented by graph nodes

$$O = \{O_n^h\ :\ n = 1,\ldots,M_h,\ h = 1,\ldots,H\},$$

the arcs are defined as

$$L = \{l_{m,n}^{g,h}\ :\ m,n = 1,\ldots,M_h,\ m \neq n,\ g,h = 1,\ldots,H,\ g \neq h\}.$$

The arcs connect subsystems which have common elements of the fault set

$$l_{m,n}^{g,h} \in L \Leftrightarrow [F_m^g \cap F_n^h \neq \emptyset],$$

the common part of these sets is denoted by

$$F_{m,n}^{g,h} = F_m^g \cap F_n^h,\ g \neq h.$$

The initial diagnosis for subsystem n at level h is defined as:

$$DGN_n^{h*} = \{f_k \in F_n^h :\ \forall_{j:s_j \in S_n^h}[v_j(f_k) = v_j]\}$$

A final diagnosis is different when detected faults belong only to the subsystem, and different when detected faults belong also to other subsystems. Set F_n^{hW} includes only faults detected in the subsystem:

$$F_n^{hW} = F_n^h - \bigcup_{l_{m,n}^{g,h} \in L} F_{m,n}^{g,h}$$

- If $DGN_n^{h*} \subseteq F_n^{hW}$ (faults are not confirmed) then the final diagnosis is equal to the initial diagnosis $DGN_n^h = DGN_n^{h*}$,
- If $DGN_n^{h*} \not\subset F_n^{hW}$ (some faults confirmed) then initial diagnosis can be specified or verified on the basis of diagnoses in other subsystems:
  - $F_{m,n}^{g,h} \neq \emptyset \wedge DGN_n^{h*} \cap DGN_m^{g*} = \emptyset \Rightarrow DGN_n^h/_m^g = \{f_k \in F_n^{hW}\}$
  - $F_{m,n}^{g,h} \neq \emptyset \wedge DGN_n^{h*} \cap DGN_m^{g*} \neq \emptyset \Rightarrow DGN_n^h/_m^g = DGN_n^{h*} \cap DGN_m^{g*}$

The diagnosis for a whole system is a collection of final diagnoses obtained from all subsystems.

The biggest advantage of this approach is decentralization of diagnostic process (faults detection) on a few computer units what results in significant lowering of calculation time. Moreover, units become independent in local diagnosis and can be used without central structure.

The biggest disadvantages are the same as for one–level diagnostic matrices. Moreover, if a fault can be detected in two subsystems, then its diagnostic signals should be always active when the fault happens. But, diagnostic signals in real systems can be compensated or noisy, so a fault can be detected in the first subsystem and not detected in the second one.

Chapter 3

Causal AND/OR/NOT graphs

Casual relations model influences of causes on observed system reactions. This kind of diagnostic description can be also considered as one of model approaches where the model is based on expert knowledge. The general idea for failure detection is based on the notion of *expected behavior*. A basic underlying assumption is that the experts have a more or less precise idea of what system behavior they can expect, i.e. they can qualify the observed behavior as the *normal* and *abnormal* one. Further, they are assumed to be able to recognize different types of abnormal behaviors and classify the observed ones into specific categories. Thus the notion of expected behavior includes both *expected normal behaviors* and various sorts of *expected abnormal behaviors*. Behaviors are described by relational models.

The main concept of causality is presented for example in (de Kleer and Brown, 1986; Iwasaki and Simon, 1986). The graphical representation of causal relations, AND/OR/NOT graphs, are described for example in (Ligęza and Fuster-Parra, 1997).

3.1 AND/OR/NOT graphs. Basic definitions

Generally, causal graphs are graphical representations of causal relations. A *causal relation* between nodes n and n' is an influence of the fact that n occurs on the fact that n' happens.

There can be a few kinds of causal influences, i.e. different kinds of causal relations (Console *et al.*, 1989; Console and Torasso, 1992). The three presented below causal relations refer to the „strength" of causality and incompleteness of our knowledge; they are as follows:

- symptom n causes symptom n' always when the former occurs; moreover, the occurrence of n' is bound to be caused by n. We refer to this type of causation as sufficient and necessary (NEC),
- symptom n causes symptom n' always when the former occurs, but there are several possible different symptoms causing n' as well. We shall refer to this type of causation as sufficient (SUF),

- occurrence of symptom n may cause symptom n' to occur, however there are cases when n' does not follow n. We refer to this weakest type of causation as possible (MAY).

The NEC relationship is the strongest one, it let us know the most precise information, but this kind of influence is very rare in real problems. Normally, two last cases are useful.

Now, the formal definition of causal graphs will be introduced:

Definition 3.1 *If N is a set of symptoms and Ψ is a set of functions defined on these symptoms, then a causal graph is a structure $G = (N, \Psi)$.*

The definition of causal AND/OR/NOT graphs can be formulated as follows:

Definition 3.2 *Let N denote a set of symptoms, $N = D \cup V \cup M$, where M is a set of manifestation symptoms, D is a set of elementary diagnoses or disorders, V is a set of pre-specified intermediate symptoms. Further, let E^* denote a set of relations defining causal dependencies, and E^- a set of relations defining binary negative dependencies. The AND/OR/NOT causal graph is a structure (N, E^*, E^-) satisfying the following conditions:*

- *the set of nodes of the graph is the set $N = D \cup V \cup M$ (for simplicity we do not distinguish between the graph nodes and labeling them symptoms),*
- *the causal relations given by E^* and E^- define the arcs in the graph,*
- *there is set of terminal nodes M and set of initial nodes D in the graph,*
- $D \cap V = \emptyset$,
- $D \cap M = \emptyset$,
- $M \cap V = \emptyset$,
- *every node has at least one arc going to this node and/or leaving from this node (the graph is connected),*
- *there are no loops in the graph.*

There are three types of nodes in *causal graphs*, their graphical representation is presented in Figure 3.1:

- OR node - logical disjunction of predecessors. If n_i is an OR–node, such that $(n_1, n_i), \ldots, (n_{i-1}, n_i)$, are all the pairs belonging to E^2 having n_i as the second argument; then the formula assigned to n_i is

$$n_i = n_1 \vee n_2 \vee \ldots \vee n_{i-1},$$

- AND node - logical conjunction of predecessors. If n_i is an AND–node, such that $(n_1, \ldots, n_{i-1}, n_i) \in E^i$, then the formula assigned to n_i is

$$n_i = n_1 \wedge n_2 \wedge \ldots \wedge n_{i-1},$$

- NOT node - logical negation of predecessors. If $(n_1, n_2) \in E^-$, the formula assigned to n_2 is of the form:

$$n_2 = \neg n_1,$$

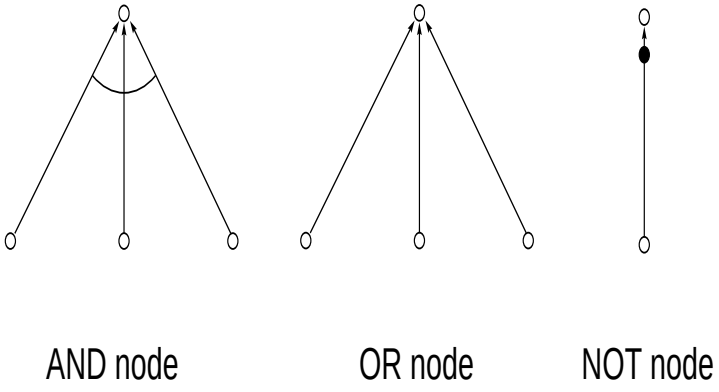

Figure 3.1: Basic nodes

Example 3.1 *The graph in Figure 3.2b presents causal relations for an simply logic AND gate* $a1$*. Structure of the graph is based on the truth table (Karnaugh, 1953) presented in Figure 3.2a. The gate has two inputs:* $aa1$ *and* $ba1$*; and one output* $ma1$*. There is only one component - the gate, and the set of potential diagnoses include* $\{a1\}$*.*

Symptoms include:

- $ma1 \in M$ *is a manifested symptom - the gate output,*
- $\{va11, va12, va13, va14, va15, va16\} \in V$ *are intermediate symptoms,*
- $da1 \in D$ *is an elementary diagnosis - a fault of the AND gate,*
- *nodes* $aa1$ *and* $ba1$ *are potential observations* OBS*.*

Assumptions constituting the bases for modeling of the causal graphs include the following:

- finite, pre-specified number of components is to be diagnosed; the undertaken control actions and operational conditions are taken into account as possible causes of abnormal behavior. The components, possible actions and conditions must be a priori known, and faulty behavior of the system may be diagnosed only by assigning faulty behavior to one or more of these components, application of certain actions and occurrence of operational conditions,

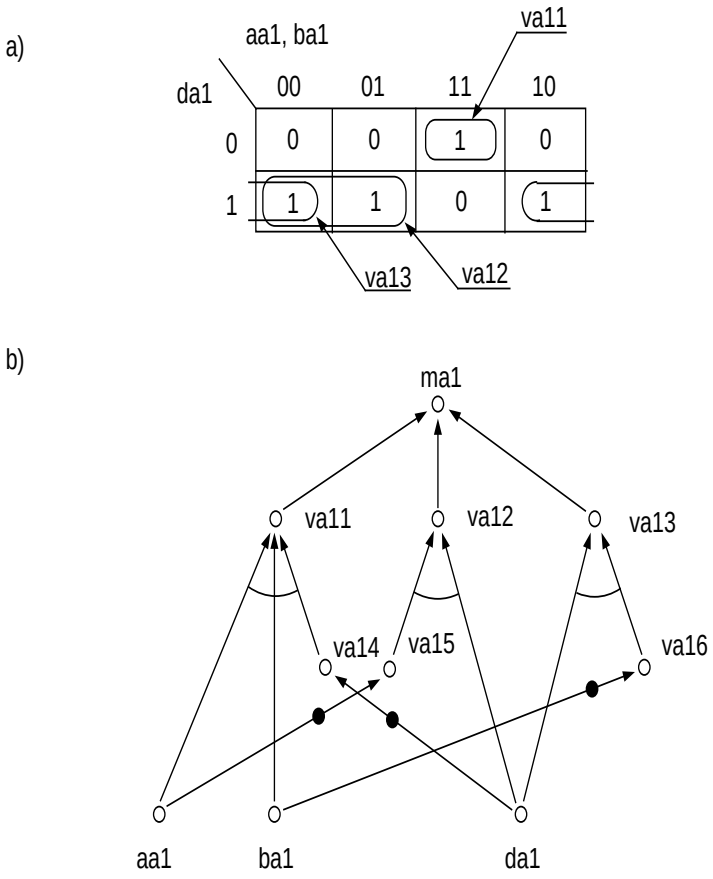

Figure 3.2: Graph AND/OR/NOT for an AND gate

- observed steady–state of the dynamic system is to be diagnosed rather than a wrong change of the state,
- the types of abnormal behavior to be diagnosed must be "predefined" in some sense, i.e. formulae describing qualitative situations referring to expected failures must be provided,
- the knowledge about causal dependencies among symptoms should be available.

3.2 The causal diagnostic methodology

The diagnostic problem for causal graphs is defined as follows:

Definition 3.3 *Diagnostic problem is defined by* (G, M, OBS)*, where* $\mathrm{M} \subseteq M$ *is a set of symptoms together with their logical values to be explained (i. e. the one specifying the abnormal behavior) and* OBS *is a set of symptoms observed to be „true" or „false" and thus providing auxiliary information for diagnostic reasoning.*

Each node $n \in N$ is characterized by a state, a logical value. The state of a node belongs to set $\{true, false, unknown\}$.

Definition 3.4 *Let* $G = (N, E^*, E^-)$ *be an AND/OR/NOT causal graph, where* $N = n_1, n_2, \ldots, n_n$*. Let* $Q = \{q^1, q^2, \ldots, q^n\}$ *be current logical values of nodes. Any set* $s = (n^1, q^1), (n^2, q^2), \ldots, (n^j, q^j)$ *will be called a state–set.*

Having defined a partial state s, one can obtain the most complete information about the status of symptoms by propagation of this state. The final state, where no further propagation is possible, will be denoted with s^*. The rules for propagation are basically the ones of logical reasoning; however, depending on the interpretation of causality various modifications are possible.

Generally speaking, all kinds of reasoning can be divided on three basic categories (Peirce, 1958):

1. Deduction, an analytic process based on the application of general rules to particular cases, with the inference of a result;
2. Induction, synthetic reasoning which infers the rule from the case and the result;
3. Abduction, another form of synthetic inference, but of the case from a rule and a result.

The diagnostic procedure for causal AND/OR/NOT graphs is an abduction procedure, abduction is backward reasoning and we can write its general rule as follows

$$\frac{\beta,\ \alpha \Rightarrow \beta}{\alpha}$$

Reasoning based on the rule allows to advance potential hypotheses which are explained conclusions. Two main abduction rules for OR and AND nodes are (Ligęza, 2003):

$$\frac{n,\ n_1 \vee n_2 \vee \ldots \vee n_i \Rightarrow n}{n_k}, \tag{3.1}$$

$$\frac{\neg n,\ n_1 \wedge n_2 \wedge \ldots \wedge n_i \Rightarrow n}{\neg n_k}, \tag{3.2}$$

where $k \in \{1, 2, \ldots, i\}$.

The presented below rules define precisely forward and backward propagation procedures for AND/OR/NOT causal graphs.

Forward propagation of graph states:

1. *OR node true*: if at least one of the predecessors of an OR node is *true*, then the value of the OR node is set to be *true*,
2. *AND node false*: if at least one of the predecessor of an AND node is *false*, then the value of the AND node is set to be *false*,
3. *NOT node true*: if a predecessor of a NOT node is *false*, then the value of the NOT node is set to be *true*,
4. *NOT node false*: if a predecessor of a NOT node is *true*, then the value of the NOT node is set to be *false*.
5. *OR node false*: if all the predecessors of an OR node are *false*, then the value of this OR node is set to be *false*,
6. *AND node true*: if all the predecessors of an AND node are *true*, then the value of this AND node is set to be *true*.

Simple backward propagation of graph states:

1. *OR node false*: if an OR node is *false*, then the values of all its predecessors are set to be *false*,
2. *AND node true*: if an AND node is *true*, then the values of all its predecessors are set to be *true*,
3. *NOT node true*: if a NOT node is *true*, then the value of its predecessor is set to be *false*,
4. *NOT node false*: if a NOT node is *false*, then the value of its predecessor is set to be *true*.

Backward propagation of graph states with selection of predecessors:

1. *OR node true*: if an OR node is *true*, then the value of one its selected predecessor is set to be *true*,

2. *AND node false*: if an AND node is *false*, then the value of one its selected predecessor is set to be *false*.

The selection allows to find all possible causes of states of OR and AND nodes. Predecessors can be selected in a predefined order or by using some other information e.g. statistic or expert knowledge. These two rules implements the abductive inference rules 3.1 and 3.2.

The above rules define the principles of state propagation. Whenever a rule is applicable, a new symptom value is generated; it is next placed in the set representing the current state. In case some symptom turns out to take two inconsistent values, the initial state for propagation is considered to be inconsistent and it is not taken into account any more.

A diagnosis generated on the basis of a *causal graph* is defined as follows:

Definition 3.5 *If G is the AND/OR/NOT causal graph defining the relationship among symptoms in the analyzed system, M is a set of manifestations to be explained, and OBS is the set of observations, then a diagnosis is any minimal and consistent set D of initial symptoms and their values, such that:*

- *D explains M, i. e. assuming that G specifies the domain theory, $G \cup D \models M$,*
- *D is consistent with observations (and their consequences), i. e. if S^* denotes the maximal state obtained by propagation of the symptom states defined by $D \cup OBS$ over G, then S^* is consistent (no symptom is true and false at the same time).*
- *D is minimal.*

Causal AND/OR/NOT graphs enable user to model behaviors of diagnosed systems. Models can be either single–level (Ligęza and Fuster-Parra, 1997) or multi–levels with using vertical and horizontal hierarchization (Ligęza and Fuster-Parra, 1998).

Chapter 4

DX and FDI model–based approaches

Diagnostic knowledge in model–based approaches has a form of an explicit model of the diagnosed system. The model includes description of correct behavior of the diagnosed system. If a model is well defined, then real world observations of the system should be consistent with its predicted behavior, otherwise we have to assume that the system does not work correctly. Location of discrepancies in the model is a basic information for diagnostic procedure.

There are two basic approaches to model–based diagnosis: *FDI* (*Fault Detection and Isolation*) and *DX* (the contraction is taken from the name of *DX International Workshop on Principles of Diagnosis*). The *FDI* approaches are closer to approaches emerging from Control Theory and statistical analysis e.g (Isermann, 1997; Patton and Chen, 1991; Frank, 1996) . The *DX* ones are closer to Computer Science and Artificial Intelligence domain (Reiter, 1987; de Kleer and Williams, 1987).

This presentation of *DX* and *FDI* model–based approaches is based, in great part, on works of Reiter and IMALAIA group (Intégration de Modéles Alliant Automatique et IA) (Cordier *et al.*, 2000). IMALAIA group designed also an interesting formalism unifing both approaches.

Example 4.1 *Further considerations about both kinds of approaches will be illustrated by simply arithmetic unit (Davis, 1984; de Kleer and Williams, 1987; Reiter, 1987). The circuit (Figure 4.1) consists of three multipliers:* $m1$, $m2$, *and* $m3$*; and two adders:* $a1$ *and* $a2$*. The values of inputs are* $A = 3, B = 2, C = 2, D = 3$*, and* $E = 3$*. The values of outputs* $F = 23$ *and* $G = 12$*. So, there is an inconsistency between the measured value of output* F *and its predicted value.*

4.1 The DX model–based diagnosis

4.1.1 Basic definitions

Reiter's diagnostic framework is based on consistency between correct behavior of a system and its observed abnormal behavior. A system is working correctly if there are no inconsistencies between these two behaviors. If some inconsistencies occur, then some parts of the system (components) are faulty. A system is defined as follows:

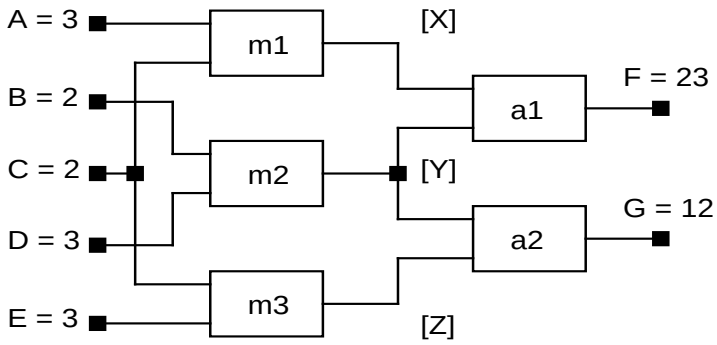

Figure 4.1: A simple arithmetic circuit.

Definition 4.1 (Reiter) *A system is a pair (SD, COMPONENTS) where:*
1) SD, the system description, is a set of first-order sentences;
2) COMPONENTS, the system components, is a finite set of constants.

Example 4.2 *Let us come back to the circuit presented in Figure 4.1. The set of components is as follows:*

$COMPONENTS = \{a1, a2, m1, m2, m3\}$,

and the system description is:

$ADD(x) \wedge \neg AB(x) \Rightarrow Output(x) = Input1(x) + Input2(x)$,
$MULT(x) \wedge \neg AB(x) \Rightarrow Output(x) = Input1(x) * Input2(x)$,
$ADD(a1)$, $ADD(a2)$, $MULT(m1)$, $MULT(m2)$, $MULT(m3)$,
$Output(m1) = Input1(a1)$, $Output(m2) = Input2(a1)$,
$Output(m2) = Input1(a2)$, $Output(m3) = Input2(a2)$,
$Input2(m1) = Input1(m3)$,

The diagnostic procedure is based on inconsistency between calculated and observed values of inputs and outputs of the components. The definition of observations is quoted below:

Definition 4.2 (Reiter) *An $OBSERVATION$ of a system is a finite set of first-order sentences. We shall write $(SD, COMPONENTS, OBS)$ for a system with observation OBS.*

Example 4.3 *Observations for the arithmetic unit are:*

$Input1(m1) = 3$, $Input2(m1) = 2$,
$Input1(m2) = 2$, $Input2(m2) = 3$,
$Input1(m3) = 2$, $Input2(m3) = 3$,
$Output(a1) = 23$, $Output(a2) = 12$.

4.1.2 Conflict and hitting sets

System description SD defines normal („correct") behavior of system components. Abnormal behavior of a component will be denoted by the predefined predicate AB. If the set $\{\neg AB(c_1), \ldots, \neg AB(c_n)\}$ represents the assumption that all the components of the system work correctly, so $SD \cup \{\neg AB(c_1), \ldots, \neg AB(c_n)\}$ represents correct behavior of the system (all components work correctly). Because of the discrepancies, the system does not work correctly. Hence, the formula given by

$$SD \cup \{\neg AB(c_1), \ldots, \neg AB(c_n)\} \cup OBS$$

is inconsistent.

A diagnosis is a conjecture that certain components of the system behave abnormally. This conjecture has to be consistent with what is known about the system and with the observations.

Definition 4.3 (Reiter) *A diagnosis for (SD, COMPONENETS, OBS) is a minimal set* $\Delta \subseteq COMPONENTS$ *such that*

$$SD \cup OBS \cup \{AB(c) \mid c \in \Delta\} \cup \{\neg AB(c) \mid c \in COMPONENTS - \Delta\}$$

is consistent.

Consistency of the system can be restored only by assumption that some of its components are faulty. The set of such components constitutes a diagnosis.

Example 4.4 *The example diagnosis for our system is* $\{m1\}$*; it means that abnormal behavior of this element explains observed misbehavior of the device.*

Finding all potential diagnoses is not easy, but they can be calculated on the basis of *conflict sets* and *hitting sets*.

Definition 4.4 (Reiter) *A conflict set for* $(SD, COMPONENTS, OBS)$ *is a set of components*
$\{c_1, \ldots, c_k\} \subseteq COMPONENTS$ *such that*

$$SD \cup OBS \cup \{\neg AB(c_1), \ldots, \neg AB(c_k)\}$$

is inconsistent.

A conflict set is any set which has at least one faulty component, i.e. a conflict set for the system is minimal iff no proper subset of it is a conflict set.

Example 4.5 *Let us come back to the example system. The prediction that Output*$(a1) = 12$ *depends on the correct operation of* $a1, m1$*, and* $m2$ *(Figure 4.2). Since Output*$(a1) = 23$ *at least one of the components* $a1, m1$*, and* $m2$ *must be faulty. Thus the set* $\{a1, m1, m2\}$ *is a conflict set. Since, none of its subset is a conflict set, the conflict* $\{a1, m1, m2\}$ *is a minimal conflict set.*

So, a diagnosis is a set which takes at least one component from each conflict set, i.e. it is a *hitting set* of all conflict sets.

Definition 4.5 (Reiter) *Suppose C is a collection of sets.*

A hitting set for C is a set $H \subseteq \bigcup_{S \in C} S$ *such that* $H \cap S \neq \emptyset$ *for each* $S \in C$.

A hitting set for C is minimal iff no proper subset of it is a hitting set for C.

Then a *hitting set* is any set having nonempty intersection with each of conflict sets and build only from elements of conflict sets. Theorem 4.1 is the basis of Reiter's theory.

Theorem 4.1 (Reiter) $\Delta \subseteq COMPONENTS$ *is a diagnosis for* $(SD, COMPONENTS, OBS)$ *iff* Δ *is a minimal hitting set for the collection of conflict sets for* $(SD, COMPONENTS, OBS)$.

Example 4.6 *For the arithmetic system, minimal conflict sets are:* $\{a1, m1, m2\}$, $\{a1, a2, m1, m3\}$.

Conflict $\{a1, m1, m2\}$ *is obvious (Figure 4.2). One can easily calculate the outputs of multipliers* $m1$ *and* $m2$ *equal* 6. *So, the output of adder* $a1$ *should equal* 12, *but the observation equals* 23 *and there is a conflict.*

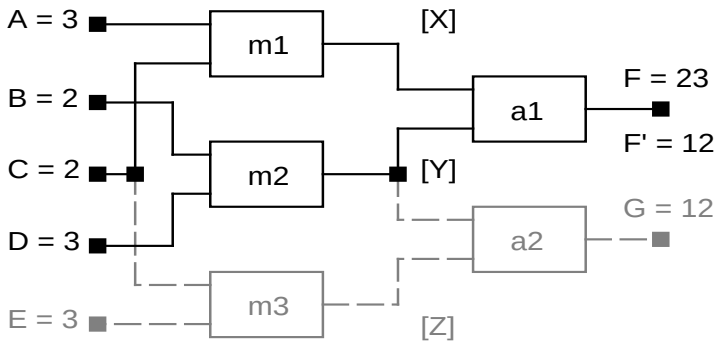

Figure 4.2: The conflict set $\{a1, m1, m2\}$.

Conflict $\{a1, a2, m1, m3\}$ *is calculated by use backward calculation (Figure 4.3). The output of multipliers* $m1$ *and* $m3$ *equal* 6. *So, we know calculated value of variable* Z *and observed value of variable G. Now, the value of variable* Y *is calculated by use adder* $a2$ *and equals* 6. *The output of adder* $a1$ *should equal* 12, *but the observation equals* 23 *and there is a conflict.*

The following minimal hitting sets can be calculated (Figure 4.4):

$d_1 = \{a1\}, d_2 = \{m1\}, d_3 = \{m2, m3\}, d_4 = \{a2, m2\}$.

Each one from the hitting sets is the diagnosis because assuming that its components are faulty allows to restore consistency of the system.

The main disadvantage of *DX* approaches is high computational complexity and because of it diagnosis of large systems becomes very hard. Practical applications of this approach are:

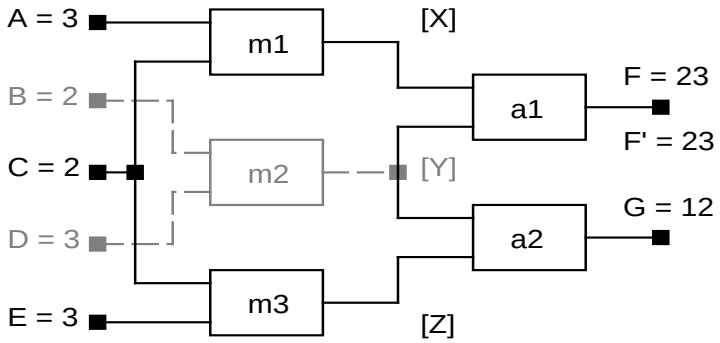

Figure 4.3: The conflict set $\{a1, a2, m1, m3\}$.

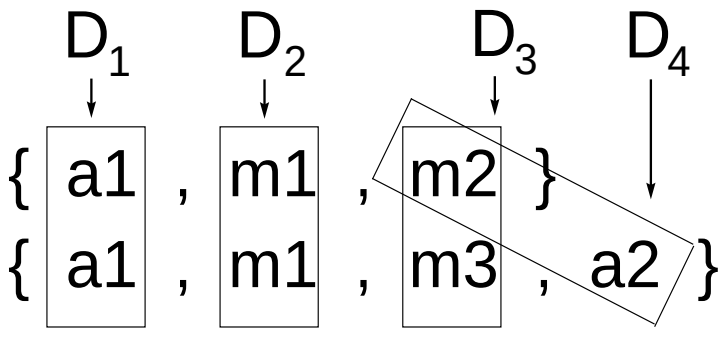

Figure 4.4: The minimal hitting sets.

- DART (Genesereth, 1984) - diagnostic methodology for logical gates.
- DEDALE (Dague *et al.*, 1987) - testing electronic circuits.
- TIGER (Milne and Travé-Massuyés, 1997) - monitoring and diagnosis of gas turbines. CA–EN module is a model–based diagnostic module.

The model–based diagnostic procedure is an incremental procedure. Diagnoses are calculated during the first stage, some localized malfunctions are verified and repaired during the second stage. Afterwards, if new acquired observations are still not consistent with calculations, the procedure is repeated.

4.2 The FDI model–based diagnosis

4.2.1 Basic definitions

Behavioral model is the main element of *FDI* approaches. The model is derived, in most cases, from system structure and shows links between system components and their model of behavior.

Definition 4.6 (Cordier et al.) *The system model SM is defined as the behavioral model BM, i.e. the set of relations defining the system behavior, together with the observation model OM, i.e. the set of relations between the variables X of the system and the observed variables O acquired by the sensors.*

Example 4.7 *Elementary components are: the adders $a1$, $a2$; the multipliers: $m1$, $m2$, $m3$. The system model SM is hence given by the following equations:*

- *BM:*
  - *Rm1: x = a * c,*
  - *Rm2: y = b * d,*
  - *Rm3: z = c * e,*
  - *Ra1: f = x + y,*
  - *Ra2: g = y + z.*
- *OM:*
  - $RSa : a = A$,
  - $RSb : b = B$,
  - $RSc : c = C$,
  - $RSd : d = D$,
  - $RSe : e = E$,
  - $RSf : f = F$,

– $RSg : g = G$.

Definition 4.7 (Cordier et al.) *A diagnostic problem is defined by the system model SM, a set of observations OBS assigning values to observed variables, and a set of faults F (i.e. a set of diagnoses; if we want to use Reiter's names). A fault can be seen as a set of faulty components.*

Example 4.8 *$OBS = \{A = 2, B = 2, C = 3, D = 3, E = 2, F = 23, G = 12\}$. The set of single faults is $SF = \{F_{a1}, F_{a2}, F_{m1}, F_{m2}, F_{m3}\}$. The set of faults is $F = 2^{SF}$ and a number of possible faults is equal $m = 2^{|SF|} - 1$.*

Definition 4.8 (Cordier et al.) *The system structure is defined through a binary application $s : SM \times V \rightarrow \{0, 1\}$, where $V = X \bigcup O$ is the set of variables. A element of binary application $s(rel, v) = 1$ if and only if v appears in relation rel.*

4.2.2 Redundancy relations

Redundancy relations are the basis of fault detection and localization in *FDI* approaches.

Definition 4.9 (Cordier et al.) *An analytical redundancy relation (ARR) is a relation entailed by SM which contains only observed variables, and which can therefore be evaluated from OBS. It is noted $r = 0$, where r is called the residual of the ARR. For a given OBS, the instance of the residual is noted $val(r, OBS)$, abbreviated as $val(r)$ when not ambiguous. Thus, $val(r, OBS) = 0$ if the observations satisfy the ARR.*

ARRs can be obtained from the system model by eliminating the unknown variables.

Example 4.9 *A complete matching leads to the following ARRs:*

- *ARR1: $r_1 = 0$ where $r_1 \equiv F - A \times C - B \times D$,*
- *ARR2: $r_2 = 0$ where $r_2 \equiv G - B \times D - C \times E$,*
- *ARR3: $r_3 = 0$ where $r_3 \equiv F - G - A \times C + C \times E$.*

Now, a definition of *fault signature* will be introduced.

Definition 4.10 (Cordier et al.) *Given a set $R = \{ARR_1, \ldots, ARR_n\}$ of n ARRs and a set $F = \{F_1, \ldots, F_m\}$ of m faults, the signature of a fault F_j is given by the binary vector $FS_j = [s_{1j}, \ldots, s_{nj}]^T$ in which s_{ij} is given by: $(ARR_i, F_j) \mapsto s_{ij} = 1$ if some components involved in F_j are involved in ARR_i, $(ARR_i, F_j) \mapsto s_{ij} = 0$ otherwise.*

Entry $s_{ij} = 0$ means that occurence of the fault F_j does not affect ARR_i ($val(r_i) = 0$). Otherwise, when $s_{ij} = 1$ then the fault F_j affects ARR_i ($val(r_i) <> 0$).

Definition 4.11 (Cordier et al.) *Given a set R of n ARRs, the signatures of a set F of m faults all put together constitute the so–called signature matrix.*

Table 4.1: The signature matrix

	F_{a1}	F_{a2}	F_{m1}	F_{m2}	F_{m3}
ARR_1	1	0	1	1	0
ARR_2	0	1	0	1	1
ARR_3	1	1	1	0	1

The signature matrices introduced here are very similar to diagnostic matrices presented in Chapter 2. The main philosophy is the same, but the terminology is different in some places.

Example 4.10 *In our example, the signature matrix for the set of single faults corresponding to components $a1$, $a2$, $m1$, $m2$ and $m3$ is shown in Table 4.1.*

If we want to diagnose multiple faults then it is necessary to extend the matrix (Table 4.1). The 26 additional columns $(2^m - 1)$ have a $[1,1,1]^T$ signature, except for $F_{\{a1,m1\}}$ which has a $[1,0,1]^T$ signature, and $F_{\{a2,m3\}}$ which has a $[0,1,1]^T$ signature.

The diagnostic sets in the *FDI* approach are given in terms of the faults accounted for in the signature matrix. The generation of the diagnostic sets is based on a column interpretation of the signature matrix and consists in comparing the *observation signature* with the fault signatures. This comparison is stated as a decision–making problem.

Definition 4.12 (Cordier et al.) *The signature of an observation OBS is a binary vector $OS = [OS_1, \ldots, OS_n]^T$ where $OS_i = 0$ iff $val(r_i, OBS) = 0$.*

The first step (the detection task) is to build the observation signature, i.e. to decide whether a residual value is zero or not, in the presence of noise and disturbances. It is generally stated as a statistical decision-making problem, making use of the available noise and disturbance models.

Example 4.11 *With OBS as above, $OS = [1,0,1]^T$. In the case $f = 10$ and $g = 10$, $OS = [1,1,0]^T$ and in the case $f = 10$ and $g = 14$, $OS = [1,1,1]^T$.*

The second step (the isolation task) is to compare the current observation signature with the fault signatures. A solution to this decision–making problem is to define a consistency criterion as follows:

Definition 4.13 (Cordier et al.) *An observation signature $OS = [OS_1, \ldots, OS_n]$ is consistent with a fault signature $FS_j = [s_{1j}, \ldots, s_{nj}]^T$ if and only if $OS_i = s_{ij}$ for all i.*

Definition 4.14 (Cordier et al.) *The diagnostic sets are given by the faults whose signatures are consistent with the observation signature.*

Example 4.12 *The diagnostic sets obtained for the following observation signatures are:*
$OS = [1,0,1]^T \Leftrightarrow F_{A1}$ *or* F_{M1} *or* $F_{\{A1,M1\}}$,
$OS = [1,1,0]^T \Leftrightarrow F_{M2}$,
$OS = [1,1,1]^T \Leftrightarrow any\ multiple$ *fault* $except\ F_{\{A1,M1\}}\ and\ F_{\{A2,M3\}}$.

Diagnoses based on *FDI* approach for the arithmetic system are $\{a1\}$, $\{m1\}$ and $\{a1, m1\}$. The same system diagnosed by *DX* methodology has diagnoses $\{a1\}$, $\{m1\}$, $\{a2, m2\}$ and $\{m2, m3\}$ what means a subsumption of *FDI* diagnoses. The cause of lack of two diagnoses is a strong limiting assumption made in *FDI* approach that several faults can never compensate each other i.e. *ARRs* are strongly independent. In many *FDI* applications a frequently adopted assumption is that only single fault may occur. There are many extensions and practical applications of this approach, e.g. (Kościelny and Syfert, 2000; Kościelny and Bartyś, 2000).

Part III

Hierarchical diagnosis

Chapter 5

Hierarchical diagnosis in heterogenic environments

5.1 Introduction

Modern technological systems are usually composed of many specialized parts, which are recursively composed of other ones. Such parts will be further called components, and a system build from numerous such components will be called a complex system. Nature of components can be either functional or structural. A functional component is understood as a part of a complex system defined by its role (behavior), while a structural component is a part of a complex system defined by its form. Component approaches to diagnosis are considered for example in (Genesereth, 1984; Davis, 1984), where they are based on structural division of larger systems.

Diagnosis of complex systems encounters two basic difficulties. The first one consists in frequent lack of a common kind of a diagnostic description for the entire system. Some groups of components can have different nature of work or can be made by different manufacturers what entails differences in best fitted or available diagnostic description between groups of components. Then it is not easy to choose a good common diagnostic description which satisfies all components and does not lose too much information.

The second problem is complexity of the diagnostic description. A proper diagnostic description for a complex system should include all components which can be changed during a reparation procedure. When a complex system is really large, then a diagnostic procedure for such a diagnostic description is computationally costly and can produce a lot of potential diagnoses.

The proposed diagnostic methodology for complex systems tries to deal with the mentioned difficulties in several ways.

Components can be described here by the mentioned before two kinds of logic-based knowledge representations: models and expert graphs. A model-based description allows to describe components more precisely, but some of the components do not have well developed models or their models are too complicated and useless for a time-limited diagnostic procedure. Such components can be described by graph-based descriptions collecting an expert knowledge about behaviors of components. Because of the two com-

plementary approaches, the methodology is able to describe systems which cannot be described by a single, common for all components methodology.

Complexity of diagnostic description is partially reduced by hierarchization, sometimes called abstraction or relaxation. We will call it further a hierarchical approach. Roughly speaking, original description is divided into a few levels and each level is constituted by "forgetting" some details from a lower level of hierarchy. The highest level is most general, the lowest one is most detailed and similar to original problem.

Here, the hierarchical structure incorporating various levels is created by inclusion of components in other components. If a component includes other components, then these subcomponents constitute next more detailed level of hierarchy. Each level of hierarchy contains several subcomponents and their borders are determined by either structural or functional division. Diagnoses obtained from higher levels of hierarchy can be better specified and validated on lower levels. A hierarchical description is usually, but not always (Giunchiglia and Walsh, 1989), less complicated and easier to diagnose than the original detailed description of a problem.

A diagnostic procedure for the presented above hierarchical diagnostic description allows for generation of diagnoses from the highest hierarchical level to the lowest one, i.e. from the most abstract level to the most detailed level (top–down). Produced diagnoses include elementary components, which have no subcomponents, or components which cannot be further diagnosed because of lack of diagnostic information. Additionally, diagnoses can be ordered according to probability of malfunctions and verified between levels of hierarchy (see Chapter 8).

5.2 Basic definitions and assumptions

Let there be given a set of constants *ELEMENTARY COMPONENTS* (*EC*, for short) denoting the basic components in the sense of (Reiter, 1987), i.e. ones denoted by constants for which no internal structure is ever considered. A (complex, hierarchically structured) component can be recursively defined as follows (Oleksiak and Ligęza, 2004):

Definition 5.1 (Component) *A component c is either:*

- *an* elementary component $c \in EC$, *or*
- *a complex component $c = (CD, SUBC)$, where CD is the component description (a set of first–order sentences) and $SUBC$ is the set of its subcomponents.*

Note that the above definition introduces a *tree of component structure* modeling relations among components and their subcomponents. Elementary components are leaves of the tree, they do not have any descendants. The root of the tree will be referred to as the *main component*. In fact, the main component is the *complex system* of hierarchical structure of subcomponents.

If $c = (CD, SUBC)$ is a component, then c is a *direct super-component* for any component $c_i \in SUBC$. Conversely, any $c_i \in SUBC$ is a *direct subcomponent* of c. If

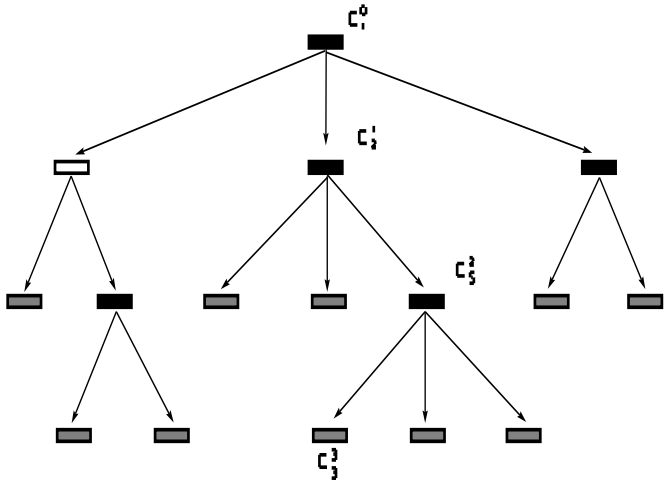

Figure 5.1: The *tree of component structure* for a hypothetical system

there exists a path, going from the root through some components c_i and c_j to some leaf, then c_i is a super-component for c_j and c_j is a subcomponent for c_i.

We further assign the *level* to any node representing some components (subcomponents) in the tree. The root node (and the main component) is assigned a *level* numbered 0. Any other component is assigned level $m + 1$, where m is the level of its direct super-component.

To denote the fact that component c_i lies at level j within component c we shall write $level(c, c_i) = j$ or $c_i \in_j c$. If component c is identified in a unique way the notation can be simplified to $level(c_i) = j$ or c_i^j.

Any component c is assumed to be identified by its (unique) name or label. To denote the fact that a component c_i is a direct subcomponent of c the path-dot notation $c.c_i$ will be used. Further, if there exists a path, going from the root through some components $c_i, c_{i+1}, \ldots, c_{i+k}$ to some leaf, and the subsequent nodes are direct subcomponents of the preceding ones, the path-dot notation will be extended to $c_i.c_{i+1}.c_{i+k}$.

Any component (either an elementary one or a complex one) can be further described by a set of variables, typically defining its inputs and outputs. It is assumed that any component has assigned some *functional behavior* and its outputs are related to inputs in some way defined by a *model* of the component. The input and output variables, together with some other characteristics of the component (internal state variables, parameters, etc.) are also referred to as *component attributes*.

To denote a value of a component attribute the standard *O–A–V (Attribute(Object) = Value)* notation will be used e.g. *output(a1) = 12*. The set ATR_i^j includes all attributes with values of component c_i^j. An undefined value is denoted by constant value "null".

In the proposed approach, combining *model-based* diagnostic procedures (Reiter, 1987) and ones based on causal reasoning with AND/OR/NOT logical causal graphs (Fuster-

Parra, 1996), any component can be perceived from two different perspectives: it can be identified as an element of the system considered at certain level in the *model-based approach* (Reiter, 1987) or in the hierarchical causal graph in the *expert approach* (Ligęza and Fuster-Parra, 1997).

The hierarchical model, similarly to the flat model, has to involve real world *observations*. The definition of observations is as follows:

Definition 5.2 *An observation of a complex component c_i^j is a finite set of first-order sentences. We shall write $(CD_i^j, SUBC_i^{j+1}, OBS_i^j)$ for a component with observation OBS_i^j.*

Example 5.1 *The arithmetic unit c_1^0 presented in figure 4.1 is modeled hierarchically. Level 0 is described by model CD_1^0 presented in the figure and includes subcomponents $SUBC_1^1 = \{m1, m2, m3, a1, a2\}$. Observations are $OBS_1^0 = \{A = 3, B = 2, C = 2, D = 3, E = 3, F = 23, G = 12\}$.*

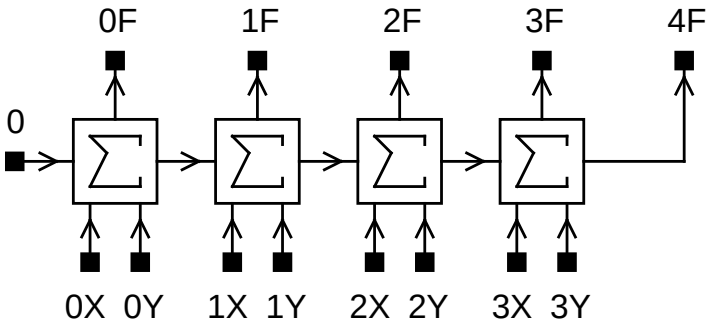

Figure 5.2: Full adder

One of the diagnoses of the system is $d_1 = \{a1\}$. Component description CD_{a1}^1 of adder $c_1^0.a1^1$ is presented in Figure 5.2, its subcomponents (one-bit adders) are $SUBC_{a1}^2 = \{\Sigma_0, \Sigma_1, \Sigma_2, \Sigma_3\}$.

5.3 Diagnostic problem

Current observations can be *consistent* with predicted behavior of a component – and in such a case the component is believed to work correctly; this is denoted as $\neg AB(c_i)$ (recall that $AB(c_i)$ denotes the fact that component c_i behaves in an *abnormal* way). If current observations are different from what can be expected on the basis of the component description (its model), the component is assumed to be *faulty* – at least one subcomponent of it behaves in *abnormal* way; this is denoted as $AB(c_i)$. The diagnostic problem for the hierarchical model can be formulated as follows:

Definition 5.3 (Diagnostic Problem) *A diagnostic problem P_i^j for component c_i^j defined at level j is defined as a four-tuple*

$$P_i^j = (c_i^j, CD_i^j, SUBC_i^{j+1}, OBS_i^j)$$

where CD_i^j is the model definition for component c_i^j, $SUBC_i^{j+1}$ are its subcomponents and OBS_i^j are the current observations at level j for component c_i^j.

Example 5.2 *The diagnostic problem for the arithmetic unit at level 0 is specified as follows:*

$$P_1^0 = (c_1^0,\ CD_1^0,\ SUBC_1^1,\ OBS_1^0).$$

Note that in the hierarchical approach, the diagnostic problem definition is always formulated at a certain *level of abstraction* and refers to some specific component. At level 0 the whole system is to be diagnosed, and the diagnoses refer to its direct subcomponents; in fact, for $j = 0$ we have the classical single-level diagnostic problem as defined in (Reiter, 1987). At some intermediate level j diagnoses of component c^j refer to its direct subcomponents.

5.4 Inter-level observations mapping

For enabling real hierarchical diagnosis one must define a way of moving a level down, so that information concerning level j can be efficiently assigned to components of level $j+1$. Two mapping functions for hierarchical systems are defined:

- **focus function** F allowing to *focus* observations on subcomponent attributes,
- **hierarchical function** H allowing to map attributes to subcomponent observations.

Definition 5.4 (Focus function) *A focus function F_i^j is a function mapping observations referring to a component c_i^j at level j to attributes referring to its direct subcomponents at level $j+1$*

$$F_i^j\colon OBS_i^j \xrightarrow{P} ATTRIBUTES_i^{j+1},$$

where $ATTRIBUTES_i^{j+1}$ is a collection of sets ATR_d^{j+1}. Set ATR_d^{j+1} includes attributes of subcomponents included in diagnosis d. Diagnosis d solves diagnostic problem P_i^j.

Example 5.3 *Let us consider again the model of arithmetic unit in Figure 4.1. The unit is a model–described one, and its attributes are inputs and outputs of its subcomponents. Four diagnoses are calculated: $d_1 = \{a1\}$, $d_2 = \{m1\}$, $d_3 = \{a2, m2\}$, $d_4 = \{m2, m3\}$.*

Collection $ATTRIBUTES_1^1$ includes four sets of attributes: $ATR_{d_1}^1$, $ATR_{d_2}^1$, $ATR_{d_3}^1$, $ATR_{d_4}^1$. For example, $ATR_{d_1}^1$ includes following attributes:

- *input1(a1) = 6, input2(a1) = 6, output(a1) = 23.*

Definition 5.5 (Hierarchical function) *A hierarchical function H_k^{j+1} is a function mapping attributes referring to a subcomponent c_k^{j+1} at level $j+1$ to its observations*

$$H_k^{j+1}\colon ATR_k^{j+1} \rightarrow OBS_k^{j+1}.$$

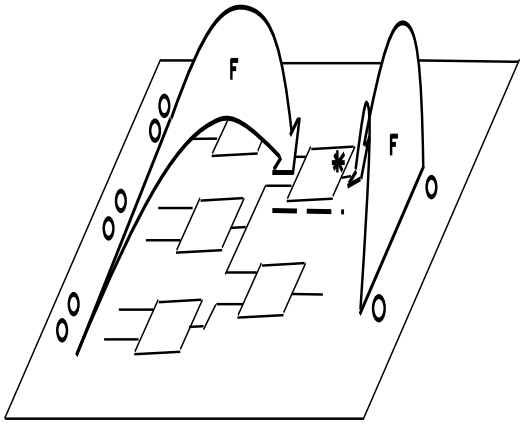

Figure 5.3: A focus function

Hierarchical function is mainly used for conversion of a form of diagnostic information between hierarchical levels. The conversion is possible if

$$\exists_{d \in P_i^j \wedge c_k^{j+1} \in d} ATR_k^{j+1} \subseteq ATR_d^{j+1}.$$

Example 5.4 *The hierarchical function H_{a2}^1 for adder $a2$ converts $\{input1(a2) = 6$, $input2(a2) = 6$, $output(a2) = 23\}$, which is a subset of the set of attributes $ATR_{d_3}^1 \in ATTRIBUTES_1^1$, to observations OBS^{j+1}. The observations are binary values on inputs and outputs of lower level components $SUBC_k^2$ (Figure 5.2):*

- $0X = 0$, $1X = 1$, $2X = 1$, $3X = 0$,
- $0Y = 0$, $1Y = 1$, $2Y = 1$, $3Y = 0$,
- $0F = 1$, $1F = 1$, $2F = 1$, $3F = 0$, $4F = 1$.

Let c_i^{j+1} be some direct subcomponent of c_k^j. The observations referring to c_i^{j+1} will be denoted as OBS_i^{j+1}, and there is $OBS_i^{j+1} \in H \circ F(OBS_k^j)$.

If subcomponent c_k^{j+1} is diagnosed as faulty on the basis of observations OBS_i^j at level j, then observations OBS_k^{j+1} resulting from mapping the observations down to level $j+1$ can fall into one of the following three categories:

- observations which are consistent with description of correct behavior of subcomponent c_k^{j+1}; the description of incorrect behavior of component c_i^j is too general. This case is similar to the TC abstraction (see Appendix B). The diagnosis is refused as false,

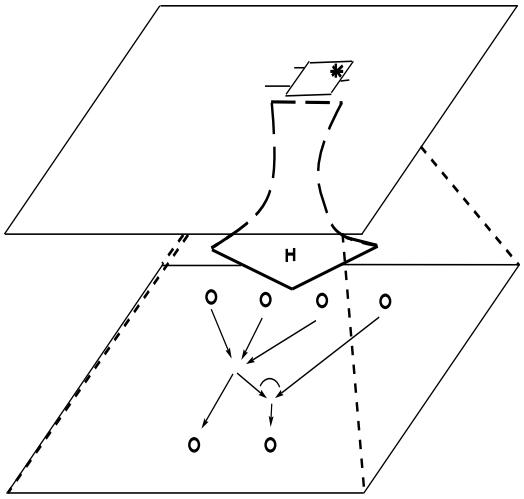

Figure 5.4: A hierarchical function

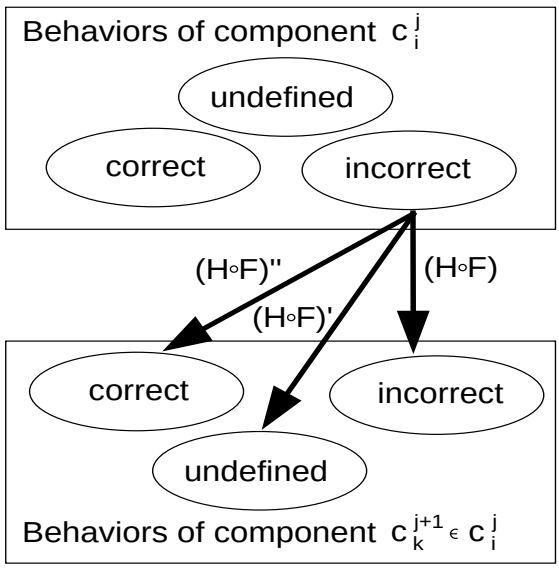

Figure 5.5: Fault mapping

- observations which are consistent with description of incorrect behavior of subcomponent c_k^{j+1}. In this case the diagnostic procedure can be continued on lower levels of hierarchy,
- observations which are not consistent with any one from the descriptions. The descriptions are incomplete, the suggested solution is to stop diagnostic procedure at this level.

The particular situations are illustrated in Figure 5.5.

Recapitulating, the form of the focus function depends on the type of component description CD^j, a form of hierarchical function depends on both: the type of component description CD^j and the type of subcomponent description CD^{j+1}.

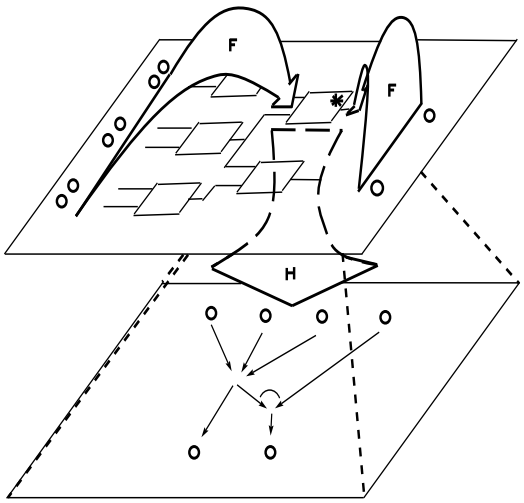

Figure 5.6: A hierarchical and a focus functions

Figure 5.6 presents both of the mapping functions in an intuitive, graphical form.

Chapter 6

Inter–level mapping for consistency reasoning and casual graphs

6.1 The focus function

6.1.1 Model–described components

Attributes for model–described levels are inputs and outputs of subcomponents. The subcomponents, which are considered in this section, have only one output. This assumption simplifies considerations without losing their generality (see the next chapter).

Values of attributes depend on: the model, observations and the diagnosis. Calculation of variables is done by the following simple algorithm:

1 Label as *"CALCULATED"* all inputs and outputs of subcomponents which have observations assigned to them. Label as *"CALCULATED"* all other inputs and outputs which are directly connected with these inputs and outputs.

2 If all inputs of a subcomponent are labeled as *"CALCULATED"*,
AND the subcomponent does not belong to the considered diagnosis,
AND the output of the subcomponent is not labeled as *"CALCULATED"* then:

- calculate value of the output of the subcomponent,
- label the output as *"CALCULATED"* and delete this subcomponent from the model,
- label also all other inputs and outputs which are directly connected with the output as *"CALCULATED"*.

3 If a subcomponent was deleted in the previous step then repeat the previous step; otherwise proceed.

4 Choose a subcomponent which does not belong to the considered diagnosis
AND it is possible to calculate some values of its inputs, then:

- calculate values of the inputs,
- label calculated inputs as *"CALCULATED"* and delete this subcomponent from the model,
- label also all other inputs and outputs which are directly connected with the inputs as *"CALCULATED"*.

5 If a subcomponents was deleted in the last cycle then go to step *2*; otherwise finish.

If we assume that faults are generated by a minimal number of faulty subcomponents from conflict sets (minimal hitting sets) then the above algorithm guarantees that all inputs and the output of each subcomponent will be calculated without any additional measurements. Otherwise, values of some inputs and outputs of subcomponents cannot be determined and the values have to be completed by additional observations. Note that further analysis of subcomponents on lower hierarchical levels is, in most cases, impossible without values of all attributes (values of inputs and outputs of subcomponents).

The above assumption allows to carry out diagnosis without engaging additional measurements and tests. So, it is useful for components where it is impossible to get any further information about subcomponents than values of inputs and outputs of the components. Moreover, diagnoses being minimal hitting sets are usually consistent with real reasons of malfunctions; so, there is high probably that such diagnoses are correct.

Now, we will try to prove the two properties of diagnoses being minimal hitting sets:

1. All inputs and outputs for each subcomponent can be calculated for such diagnoses.
2. Outputs of each faulty subcomponent can be calculated and the resulting values will be inconsistent with the ones expected on the basis of the model of correct behavior of the subcomponent and its observed inputs.

The first property of diagnoses is that all inputs and outputs of each subcomponent will be calculated if diagnoses are minimal hitting sets. The problem with calculation can appear only when a subcomponent is recognized as faulty and all its „ways of calculation" (conflict sets) include some other faulty subcomponents. Then some outputs or inputs of the subcomponent are undefined.

Lemma 6.1 *If hitting set h_j calculated from a collection of conflict sets $CON = \{con_1, \ldots, con_k\}$ is minimal, then*

$$\forall_{c_1 \in h_j} \forall_{c_2 \in h_j : c_2 \neq c1}$$
$$\Pi(c_1) - \Pi(c_2) \neq \emptyset \;\; and \;\; \Pi(c_2) - \Pi(c_1) \neq \emptyset,$$

where $\Pi(c)$ is a collection of conflict sets such that

$$\Pi(c) = \{con \in CON \colon c \in con\}.$$

Proof 6.1 *Assume that hitting set h_j is minimal and there exists a pair of subcomponents c_1 and c_2 in a hitting set h_j such that*

$$\Pi(c_1) - \Pi(c_2) = \emptyset.$$

It means that

$$\Pi(c_1) \subseteq \Pi(c_2)$$

and the set

$$h'_j = h_j - \{c_1\}$$

is also a hitting set. Hence

$$h'_j \subset h_j$$

and the hitting set h_j is not minimal.

Example 6.1 *Let the hitting set h_j be defined as follows:*

$$h_j = \{c_1, c_2, c_6\},$$

when conflict sets are:

$$con_1 = \{c_1, c_2, c_3\},$$
$$con_2 = \{c_1, c_2, c_4\},$$
$$con_3 = \{c_2, c_6\},$$
$$con_4 = \{c_6, c_7\}.$$

So,

$$\Pi(c_1) = \{con_1, con_2\},$$
$$\Pi(c_2) = \{con_1, con_2, con_3\},$$

and

$$\Pi(c_1) \subseteq \Pi(c_2).$$

Subcomponent c_2 is included in the same conflict sets as c_1 and subcomponent c_1 can be removed from h_j and the set will be still a hitting set.

$$h'_j = h_j - \{c_1\} = \{c_2, c_6\}.$$

Moreover, hitting set h'_j is a minimal hitting set.

Theorem 6.1 *If diagnosis d_j is a minimal hitting set of a collection of conflict sets $CON = \{con_1, \ldots, con_k\}$ then*

$$\forall_{c_i \in d_j} \ \exists_{con \in CON} \quad con \cap d_j = \{c_i\}.$$

Each subcomponent in a diagnosis, which is a minimal hitting set, belongs to a conflict set such that there are no other faulty subcomponents from the diagnosis.

Proof 6.2 *The proof results directly from lemma 6.1. If we take a subcomponent $c_s \in dj$, then it has to belong to at least one conflict set such that no other subcomponent from d_j belongs to.*

So, inputs and outputs of such subcomponent can be calculated on the basis of the conflict set. We can describe the conflict set with use of a set of equations which are inconsistent. When one of the equations, representing the subcomponent is removed from the set of equations, then the set of equations becomes consistent. Moreover, they are divided into two consistent and coherent groups.

The first group includes some equations and only one undefined variable - the value of output of the subcomponent. The value can be calculated (as far as backward calculation is possible) if a number of equations is greater than or equal to 1, what is true for this group.

The second group includes some equations and as many undefined variables as the number of inputs of the subcomponent. The values can be calculated because they were obtained during refutation procedure which generated the conflict set. Therefore, now, such calculation is also possible.

Example 6.2 *Let the conflict sets are:*

$$con_1 = \{c_1, c_2\},$$

$$con_2 = \{c_2, c_3\},$$

$$con_3 = \{c_3, c_4\}.$$

One of the minimal diagnoses is

$$d_1 = \{c_2, c_3\}.$$

Values of inputs and outputs of subcomponent c_2 can be calculated on the basis of conflict set con_1, subcomponent c_3 can be calculated on the basis of con_3.

The second property of diagnoses, which are minimal hitting sets, is that values of outputs of faulty subcomponents can be calculated (see Theorem 6.1) and they are not consistent with description CD (correct behavior) of the subcomponents.

If a diagnosis is not minimal hitting set then additional measurements are necessary. The measurements have to precise values of inputs and outputs of subcomponents which are undefined or are calculated on the basis of sets of subcomponents which are either not superset or not equal to some conflict sets.

Theorem 6.2 **If** *diagnosis d_j is a minimal hitting set* **and** *subcomponent c_s is a part of this diagnosis,* **and** *an output of subcomponent c_s can be calculated on the basis of other subcomponents and observations* **then** *the calculated value will be different than the expected correct behavior of subcomponent c_s.*

Example 6.3 *Let us consider again the model of arithmetic unit in Figure 4.1. Diagnosis $d_2 = \{m1\}$ is a minimal hitting set of conflict sets: $con_1 = \{a1, m1, m2\}$, $con_2 = \{a1, a2, m1, m3\}$. Then calculated values of inputs and outputs are:*

- $output(m1) = 17$,
- $input1(m1) = 3$,

- $input2(m1) = 2$.

Diagnosis $d_2' = \{m1, m3\}$ is also a hitting set and a diagnosis. The calculated values are:

- $output(m1) = 17$,
- $input1(m1) = 3$,
- $input2(m1) = 2$,
- $output(m3) = 6$,
- $input1(m3) = 3$,
- $input2(m3) = 2$.

The behavior of component $m3$ is quite correct and diagnosis d_2' will be refused on lower level of hierarchy.

Proof 6.3 *Assume that the above theorem is not true. Then it is possible to calculate output of faulty subcomponent c_s basing on a set of subcomponents K and the calculated value is the same as the correct behavior of subcomponent c_s for specified inputs. Each conflict set has to include at least one subcomponent that the calculated value of its output is different than the observed one. Then, according to Theorem 6.1, there exists a conflict set $con_n \in CON$ such that the subcomponent is the only one faulty subcomponent of it. The conflict can be calculated basing on the correct calculated output of the faulty subcomponent and the rest of subcomponents in the conflict set. Then the set*

$$con_n' = K \cup P,$$

where

$$P = con_n - ANTECEDENTS(c_s) - \{c_s\},$$

is a conflict set.

The function $ANTECEDENTS(c_i^j)$ returns a set of components that all belongs to path $c_1^0 \ldots . c_i^j$, component c_i^j does not belong to the set.

- *If $\neg\exists_{\ con_n'' \in CON}\ \ con_n' \subset con_n'' \ \vee\ con_n' \supseteq con_n''$, then collection of conflicts CON is not a complete family of conflict sets.*
- *If $\exists_{\ con_n'' \in CON}\ \ con_n' \subset con_n''$, then con_n'' is not a minimal conflict set.*
- *If $\exists_{\ con_n'' \in CON}\ \ con_n' \supseteq con_n''$, then diagnosis has to include at least one subcomponent c_s'' from con_n''*
  - *If c_s'' belongs to the set K, then the subcomponent cannot be calculated based only on the correct observations.*

– *If c_s'' belongs to the set P, then con_n contents more than one faulty subcomponent and the hitting set is not minimal.*

Recapitulating, if diagnoses are minimal hitting sets then all inputs and outputs of faulty subcomponents can be calculated and the diagnostic procedure can be continued on lower levels of the hierarchical model. This can be done without additional measurements.

6.1.2 Graph–described components

Observations and intermediate symptoms, taking part in diagnosis of a graph–described subcomponent s_p, become directly attributes of the subcomponent.

Let the list H_d include history of state propagation in a causal graph G for a diagnosis d. Then it is possible to determine all nodes which take a part in creation of diagnosis d and they are either manifestation symptoms or intermediate symptoms. The nodes (symptoms) constitute a set P_d and they become attributes of each subcomponent included in diagnosis d.

Example 6.4 *Let us consider a graph–described printer feeder (Figure 6.1). Component $Feeder$ is at level 1. Its subcomponents are at level 2: $Feeder\ drive$, $Feeder\ rolls$, $Paper$, $Holding\ unit$. One of the diagnoses is $d_1 = \{Feeder\ drive\}$.*

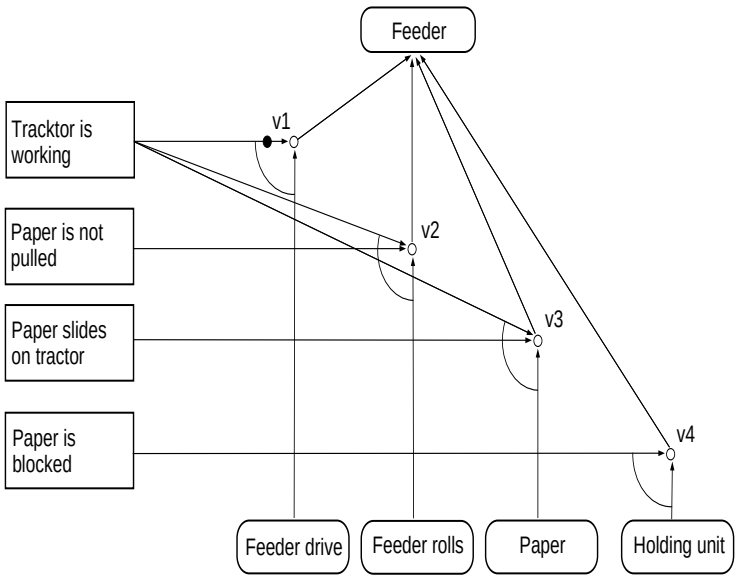

Figure 6.1: Diagnostic description for a printer feeder

In Figure 6.1, the names in ovals refer to elementary diagnoses (faults) of components; the symptoms in rectangles define auxiliary observations.

All symptoms from set P_{d_1} become attributes of subcomponent $Feeder\ drive$, the set includes the following symptoms:

- $v1$,
- $Tractor\ is\ working$,
- $Feeder$.

Graph description has one unpleasant property: a lot of diagnostic information is lost during mapping through graph–described components.

Example 6.5 *Let component c^j be a model–described component, c^{j+1} be a graph–described component, c^{j+2} be a model–described component and $c^j.c^{j+1}.c^{j+2}$.*

Component c^{j+2} is model-based and needs a precise set of observations e.g. $input(m1) = 3$. But causal graphs are mainly based on qualitative observations e.g. $valve\ is\ open$, $Temp. > 50$. Such information is more convenient for an expert who, in most cases, is a source of knowledge codified in the graphs.

Then the precise information from level j is converted to qualitative information at level $j + 1$ and further it has to be again converted to precise information at level $j + 2$. The last convertion is, in most cases, impossible without additional information (e.g. measurements).

Deficit of information is especially harmful when subcomponents are described by their models. If values of some attributes are unknown or not precisely evaluated then not all the conflicts are generated for model–described subcomponents, since some ways of calculation of conflicts become unavailable. When not all conflict sets are generated then not all diagnoses are generated, moreover some diagnoses can be faulty.

Example 6.6 *A hypothetical model–described component is diagnosed twice:*

- *There is one conflict set: {$c1$, $c2$}. The minimal hitting sets are: {$c1$}, {$c2$}.*
- *There are two conflict sets: {$c1$, $c2$}, {$c2$, $c3$}. The minimal hitting sets are: {$c2$}, {$c1$, $c3$}.*

Hitting set {c1}, which is generated in the first case, is a false diagnosis in the second case. Of course, the component $c1$ is considered as faulty, but the real faults cannot be generated only by this single component.

It is possible to carry out such diagnosis but it is also necessary to be aware that set of diagnoses can be incomplete. We can propose a few approaches to keep information flow among the distinguished levels of hierarchy at a satisfactory level of precision:

- additional measurements and interaction with users,
- guaranteeing that there are no model–described components below graph–described components in the hierarchical model,
- parameterization of subcomponents by typical values for certain kinds of faults,

- use of some additional knowledge holders for graph–described components.

The last case is interesting. It can be performed by use of a model, with information about connections among subcomponents, which is associated to the graph. The model includes information about proper behavior of some subcomponents, similar to model–described components, but there are also subcomponents without such information. Such kind of diagnostic description will be called an extended graph–description; a model associated with an expert graph will be called an auxiliary model.

Calculation of additional attributes for the extended graph–description is simple. Diagnoses are generated on the basis of the expert graph. Additionally, all diagnoses obtained from the graph have to be consistent also with the auxiliary model and observations.

The presented below algorithm checks diagnoses according to consistency with auxiliary models.

```
1 Delete subcomponents which are part of the diagnosis.
  Delete subcomponents without proper descriptions of
  behavior.

2 Calculate values of inputs and outputs of subcomponents
  in the model as long as it is possible.

3 If some calculated values are inconsistent with some ob-
  servations then the diagnosis is also inconsistent with
  the model.  Otherwise, the diagnosis can be accepted.
```

After that, it is possibility to calculate values for some inputs/outputs of faulty subcomponents on the basis of the model. The calculation can be done by an algorithm which is similar to the presented before one for model–described components.

6.2 The hierarchical function

The hierarchical function maps attributes to observations. The role of this function is usually auxiliary and reduced to conversion of diagnostic information from one form to another. We can distinguish here four possible configurations of two hierarchical levels and two types of description:

- Model–model configuration. The hierarchical function maps attributes which represent a subcomponent inputs or outputs to the same inputs or outputs on the lower level of hierarchy. Sometimes, it is necessary an additional conversion, e.g. an integer value is divided into bits (see Example 5.4),

- Model–graph configuration. Observations in the graph are usually propositional symbols with boolean values, subcomponent attributes are converted to observations by pre-specified rules e.g.
  IF flow(valve1) > 15 THEN valve_full_open := TRUE ELSE valve_full_open := FALSE.

- Graph–model configuration. The hierarchical function for this case depends on additional solutions used with focus function for improving mapping of diagnostic information. Two typical solutions are similar to these which were presented in previous points e.g.:
  auxiliary model: position := position(encoder),
  rules: IF NOT encoder_ok AND flow_max THEN position := 15 AND value := 30,

- Graph–graph configuration. The situation is quite similar to previous point.

Chapter 7

Diagnostic algorithm

The presented algorithm is an implementation of the hierarchical diagnostic methodology presented in Chapter 5 and Chapter 6. The algorithm is composed of a *global diagnostic procedure* and a *local diagnostic procedure*. The *global diagnostic procedure* is started with given *complex component* c_i^j and its observations OBS_i^j.

In the first step, diagnostic problem P_i^j is solved by the local diagnostic procedure. Local diagnoses obtaining during this stage are graphically represented as an *intermediate graph*. In the next step, each local diagnosis d_k is analyzed and *set of attributes* $ATR_{d_k}^{j+1}$ for the diagnosis is calculated. This part of the diagnostic procedure implements the focus function. All *sets of attributes* and the *intermediate graph* are returned to global diagnostic procedure.

The further diagnosis is driven according to depth–first strategy; breadth–first strategy is also available but is not considered in this chapter. A first local diagnosis d_1 is taken from the *intermediate graph*. Now, some first subcomponent c_p^{j+1} is chosen from diagnosis d_1 and its observations OBS_p^{j+1} are calculated on the basis of attributes $ATR_{d_1}^{j+1}$. This part of the diagnostic algorithm implements the hierarchical function.

When diagnosis is finished for subcomponent c_p^{j+1} then next component (in predefined order) is taken from diagnosis d_1 and the procedure is repeated up to the last one in the diagnosis. The same operation is done with the rest of diagnoses for component c_i^j.

Next sections present more precisely this diagnostic procedure.

7.1 The local diagnostic algorithm

The local diagnostic algorithm diagnoses components by using proper methodology for their diagnostic description CD. Result of the local diagnosis is an *intermediate graph* being a simple graph with structure specific for the type of component description (Oleksiak, 2004). The main reason for use of a graph form instead of a simple collection of diagnoses is that a graphical form is more readable for users.

Two kinds of diagnostic descriptions can be analyzed by the local diagnostic algorithm:

- causal AND/OR/NOT graphs,

- models (which are useful for consistency based reasoning).

In the first case, diagnosis on the basis of casual AND/OR/NOT graphs is done by propagation of states. The result of the diagnostic procedure for a *graph–described component* can be written as disjunction of all diagnoses for the component where each diagnosis is represented by conjunction of faulty subcomponents. The intermediate graph is built as follows:

```
root := createOrNode()
FOREACH diagnosis D_i IN diagnosisSet DO
  diagnosisNode := createAndNode()
  addNode(root, diagnosisNode)
  FOR EACH component C_ij IN D_i DO
    componentLeaf := createLeaf(C_ij)
    addNode(diagnosisNode, componentLeaf)
  END FOREACH
END FOREACH
```

The notation of graph nodes is similar to notation which is used for causal graphs. OR nodes are ordinary graph nodes; they represent logical disjunction of descendant nodes. AND nodes are nodes with an additional arc under the node; they represent logical conjunction of descendant nodes. Leaves of graphs are labeled by component names.

When it is necessary to calculate diagnoses from *an intermediate graph* then the state of its root node is set to "true". Further calculations are done according to state propagation rules in causal AND/OR/NOT graphs. If state of a leaf is "true", the component, whose name is on the label, is considered as faulty and becomes a part of a diagnosis.

Example 7.1 *The intermediate graph for a hypothetical valve is presented below, component description of the valve is a causal graph. Diagnoses for the system are: $\{f, a\}$ and $\{e\}$.*

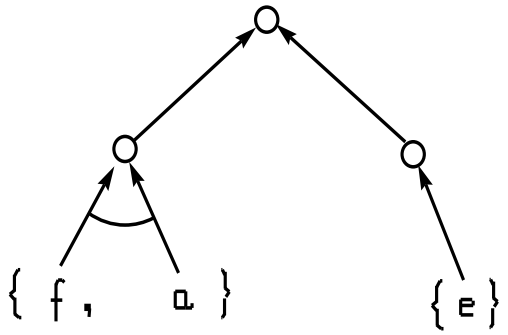

Figure 7.1: The intermediate graph for a valve

In the second case, model–described components are diagnosed by the refutation procedure. Models have to fulfill some limitations; the following assumptions are completing those included in the original theory (Reiter, 1987):

1. A proper behavior of a subcomponent is described as

$$Y = f(X),$$

where

$$X = (x_1, x_2, x_3, \ldots, x_n),$$
$$Y = (y_1, y_2, y_3, \ldots, y_k).$$

Y is a vector of output variables, *X* is a vector of input variables.

2. Function *f* can be calculated forward if all its inputs are specified. Function f^{-1} can be calculated backward for an input variable if other inputs and outputs are known.

Components with more than one output are (conceptually) split before diagnosis and they are assembled again just after it. The number of new (virtual) components resulting from the split is equal to the number of outputs in the original component. If at least one from the new components is recognized as faulty then the original component is faulty and will be further diagnosed on lower hierarchical levels (if applicable).

Diagnoses for a model–described component are generated in two stages. At the first stage, conflict sets are calculated based on component observations and the model. If there is a difference between the value of an observation and the corresponding value obtained from the model, then observations are inconsistent with the model. Refutation procedure generates all possible sets of components which have influence on inconsistencies. At the second stage, diagnoses are generated from *conflict sets*; each diagnosis is a *hitting set* of all *conflict sets*.

The collection of conflict sets is transformed to an *intermediate graph*. The transformation can be done similar to transformation for graph–described components, but it can be also done in more compact way which better explains how diagnoses are generated for *model–described components*. The last property is especially useful for interaction with users.

An *intermediate graph* is built as follows:

```
root := createAndNode()
FOREACH conflictSet C_i IN conflictSets DO
  conflictNode := createOrNode()
  addNode(root, conflictNode)
  FOR EACH component C_ij IN C_i DO
    componentLeaf := createLeaf(C_ij)
    addNode(conflictNode, componentLeaf)
  END FOREACH
END FOREACH
```

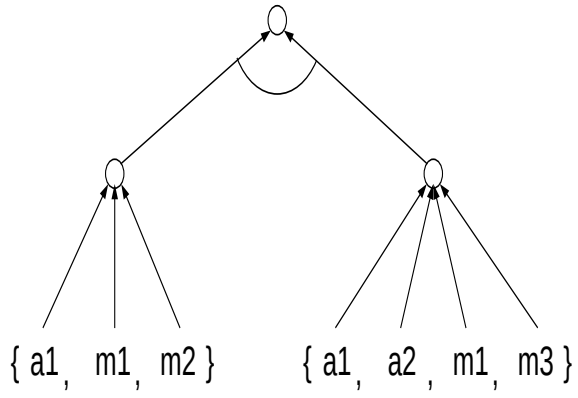

Figure 7.2: The intermediate graph for the model–described arithmetic system

Example 7.2 *The intermediate graph for the model–described arithmetic system in Figure 4.1 is presented in Figure 7.2. Conflict sets for the system are: {a1, m1, m2} and {a1, a2, m1, m3}.*

A diagnosis is any hitting set of the conflict sets. This means that it is necessary to take at least one subcomponent from each conflict set to form a diagnosis. The original methodology for generation of minimal hitting sets is presented in Appendix A.

The simple graph procedure, which propagates states in such an intermediate graph, takes one component from each conflict set. Note that some diagnoses generated in this way may be not minimal with respect to set inclusion. It was shown that minimal hitting sets have some useful properties for the hierarchical diagnosis, so, it is on purpose to bring closer diagnoses from intermediate graph to pure minimal hitting sets.

The graph procedure will take subcomponent only from these hitting sets which include none of the previously selected subcomponents (from previous hitting sets).

The rule partially eliminates hitting sets which include more components than minimum and it is very similar to the main rule for building HS–tree (Reiter, 1987).

Unfortunately, only some non-minimal hitting sets are eliminated in this way because of sequence of the selected conflict sets (sequential nature of the graph procedure). When it is necessary to calculate really minimal hitting sets, in set inclusion meaning, then every generated hitting set should be compared with already produced hitting sets and either rejected if it is a superset or accepted in place of all its supersets.

7.2 The global diagnostic algorithm

The result of diagnostic process for a complex component is a collection of alternative diagnoses. Each diagnosis is an explanation of abnormal behavior of the complex component and includes some of its subcomponents. All subcomponents in a diagnosis have to be faulty for keeping consistency between observations and complex component description CD.

The logical aspect of hierarchical diagnostic reasoning requires that the following formula is recursively satisfied:

$$AB(c_i^j) = \bigvee_{d_k \in D_i} \Big(\bigwedge_{c_n^{j+1} \in d_k} AB(c_n^{j+1}) \Big)$$

Recursion of the formula is finished when either the collection of diagnoses for a complex component is empty or a component is an elementary component. More precisely, the diagnostic procedure is stopped and returned in the form of a *tree of diagnoses* when:

1. A diagnosed component is an elementary component. The algorithm turns back to its supercomponent and the component is recognized as an atomic element of a diagnosis.

2. It is impossible to establish values for suitable number of attributes or observations for a component. The component is treated as an elementary component.

3. Diagnostic procedure is finished without any diagnosis. This happens if either observations generate a conflict in a causal graph or observations are not generating any conflict in a model. The algorithm turns back to supercomponent, the currently examined diagnosis is skipped, and the next diagnosis in order is analyzed.

4. The established values of attributes S_1 are in conflict with either domains of attributes or other conditions imposed on attributes. The algorithm turns back to its supercomponent, the current diagnosis is skipped, and the next diagnosis in order is analyzed. All sets of attributes S_2, such that $S_1 \subseteq S_2$ will be also conflicting (for both model–described and graph–described components).

The final result of diagnostic process can be shown in visual form as a simple *tree of diagnoses* T_R (e.g. Figure 7.3). Each complex component c_i^j labels a *component node*; the node is an OR node. Each diagnosis d_k of complex component c_i^j is represented by a descendant node of the *component node*. Nodes labeled by diagnoses are *diagnosis nodes* and they are AND nodes.

Each subcomponent of component c_i^j which belongs to a diagnosis is a descendant of the *diagnosis node*. *Diagnosis nodes* can be *expanded* to *component nodes* and *component nodes* can be expanded to further *diagnosis nodes* (if diagnoses exist). The highest complex component in hierarchy is represented by a root of T_R. The rest of the tree is built recursively down according to results of diagnosis of complex components.

Consider a tree modeling the hierarchical diagnostic procedure, such as one presented in Figure 7.3. Let us introduce the following definition (Oleksiak and Ligęza, 2005a,b):

Definition 7.1 *A hierarchical diagnosis for any component c_i^j at level j is any subtree satisfying the following conditions:*

- *the root of the subtree is any of the diagnosis nodes located directly under component node c_i^j,*

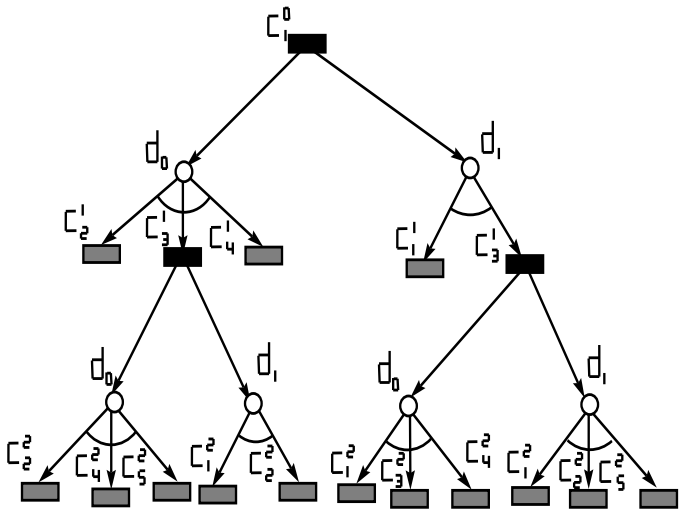

Figure 7.3: Tree of diagnoses T_R for a complex system

- *for any component node c_n^k in the tree, at most one subtree with its root node being a diagnosis node for the component node c_n^k is belongs to the tree defining the hierarchical diagnosis.*

With respect to the above definition any component in the tree modeling hierarchical diagnosis can be expanded (down) and a hierarchical diagnosis for it is developed in this way; for the leave nodes, however, no expansion is possible (they are *elementary diagnoses*). The hierarchical diagnosis for the whole system is built as a subtree satisfying above definition and with root diagnosis node located directly under component c_1^0.

Note that for a given component, various diagnoses can be defined with respect to the degree of expansion. For example, there are two different hierarchical diagnoses marked in Figure 7.4 and Figure 7.5.

The *most expanded hierarchical diagnosis* is a diagnosis whose all *component nodes* have no descendant nodes, i.e. one in which all the component nodes are in fact *elementary components* (they have no internal structure of subcomponents). But, in most cases, such diagnosis is difficult to obtain because of problems with mapping during the hierarchical diagnostic process. The diagnosis presented in Figure 7.5 ($\{c_1^1, c_1^2, c_3^2, c_4^2\}$) is a *most expanded hierarchical diagnosis* if all its components are *elementary components* EC.

The global diagnostic procedure carries on diagnosis of the whole system and the diagnosis is based on results obtained from the local diagnostic procedure. The global diagnostic algorithm is based on top–down search in the *tree of diagnoses*. Such a search has two basic versions: depth–first search and breadth–first search.

The simplest sketch of the depth–first searching procedure is as follows:

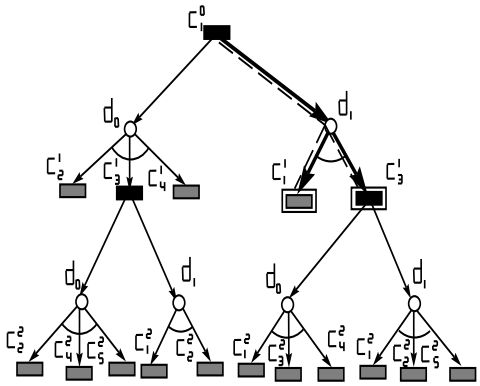

Figure 7.4: Tree of diagnoses T_R with marked hierarchical diagnosis $\{ c_1^1, c_3^1 \}$

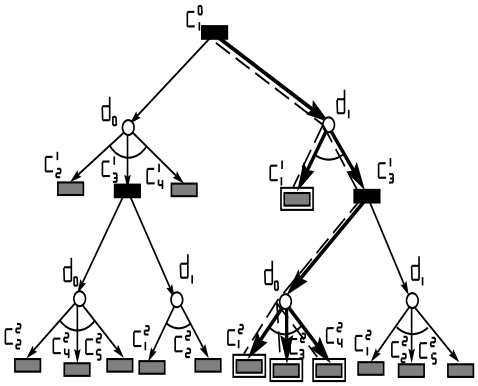

Figure 7.5: Tree of diagnoses T_R with marked hierarchical diagnosis $\{ c_1^1, c_1^2, c_3^2, c_4^2\}$

1. i = 1, j = 0
2. $P_i^j \rightarrow D_i^j$
3. SELECT p, k THAT $c_k^{j+1} \in d_p$ IS NOT PROCESSED AND $d_p \in D_i^j$
4. IF k EXISTS THEN $j = j + 1$, $i = k$ AND GOTO 2
5. IF $j = 0$ THEN EXIT
6. IF k NOT EXISTS THEN $j = j - 1$ AND GOTO 3

Detailed form of the depth–first search procedure is presented below (written in pseudo–code). It is started with a root component at level 0 as parameter and is further called recursively up to generation of all possible diagnoses.

```
PROCEDURE globalDiagnosticD(component, observations, nodeOR)
  intermediateGraph, attributesF := localDiagnosticF(
     component, observations)
  IF intermediateGraph IS EMPTY THEN
    RETURN component_refused
  END IF
  diagnosis := getFirstDiagnosis(intermediateGraph)
  WHILE diagnosis IS NOT EMPTY DO
    diagnosisNode := createAndNode()
    addNode(nodeOR, diagnosisNode)
    FOREACH subcomponent IN diagnosis DO
      subcomponentNode := createOrNode()
      addNode(diagnosisNode, subcomponentNode)
      IF subcomponent IS elementary THEN
        closeNode(subcomponentNode)
```

```
      CONTINUE
    END IF
    atr := getAttributes(diagnosis, subcomponent,
           attributesF)
    conditions := getConditionsCND(subcomponent)
    IF NOT checkConditions(atr, conditions) THEN
      EXIT FOREACH
    END IF
    subobservations:=mapAttributesH(subcomponent, atr)
    IF NOT correctSetOf(atr, subobservations) THEN
      closeNode(subcomponentNode)
      CONTINUE
    END IF
    IF globalDiagnosticD(subcomponent, subobservations,
       subcomponentNode) = component_refused THEN
      closeNode(diagnosisNode)
      EXIT FOREACH
    END IF
  END FOREACH
  diagnosis := getNextDiagnosis(intermediateGraph)
 END WHILE
END PROCEDURE
```

The diagnoses for the whole complex system can be shown when the diagnostic procedure is finished.

7.3 Computational cost of the global diagnostic algorithm

It is hard to consider the complexity of the hierarchical algorithm because it depends strongly on the used methodologies in the local diagnostic algorithm. Even more complex is the model–based diagnostic algorithm. There are a few kinds of this procedure (Darwiche and Provan, 1997) which produce: minimal, kernel (de Kleer *et al.*, 1992) or MT diagnoses (Friedrich, 1993). They have different complexity, from NP–hard for the original methodology to polynomial for some methodologies which produce only specific subsets of diagnoses for specific kinds of systems.

First, it is necessary to determine how many sets of values can be calculated for inputs and outputs of a subcomponent s which is a part of a diagnosis for a model described component.

A number of possible different sets of values for a subcomponent s is roughly limited by

$$v_s^{max} = \sum_{i=1}^{b} \binom{b}{i}$$

where b is a number of different observed variables (places) in a model where the cal-

culated values can be different than the observed ones. For further considerations, it is assumed to be equal to a number of component outputs . We can say that v_s^{max} is the most pessimistic number of sets for model–described components.

Example 7.3 *Let us modify a little bit the arithmetic unit (Figure 7.6). There are the following conflict sets for the presented observations:*

$$con_1 = \{a1, m1, m2\},$$
$$con_2 = \{a2, m2, m3\},$$

$$con_3 = \{a1, a2, m1, m3\},$$
$$con_4 = \{a1, a3, m3, m2\},$$
$$con_5 = \{a2, a3, m2, m1\},$$

$$con_6 = \{a1, a2, a3, m1\},$$
$$con_7 = \{a1, a2, a3, m2\},$$
$$con_8 = \{a1, a2, a3, m3\}.$$

There are three diagnoses, among others, which include component $m1$*:*

$$d_1 = \{m1, a2, m3\},$$
$$d_2 = \{m1, a3, m2\},$$
$$d_3 = \{m1, a1, m3\}.$$

Values at inputs of component $m1$ *are the same for all diagnoses:* 2 *and* 3*. The value at output of component* $m1$ *is different for each one from the three diagnoses:*

- d_1 *- output of* $m1$ *is equal 17,*
- d_2 *- output of* $m1$ *is equal 12,*
- d_3 *- output of* $m1$ *is equal 1.*

We have three sets of values ($v_{m1} = 3$*):*

- $\{input1(m1) = 2, input2(m1) = 3, output(m1) = 17\}$,
- $\{input1(m1) = 2, input2(m1) = 3, output(m1) = 12\}$,
- $\{input1(m1) = 2, input2(m1) = 3, output(m1) = 1\}$.

A number of unit outputs is equal 3 *but a number of outputs where there are differences between the calculated and the observed values is* 2*. Therefore,*

$$b = 2,$$
$$v_s^{max} = \sum_{i=1}^{2} \binom{2}{i} = 3,$$
$$v_s^{max} \geq v_{m1}.$$

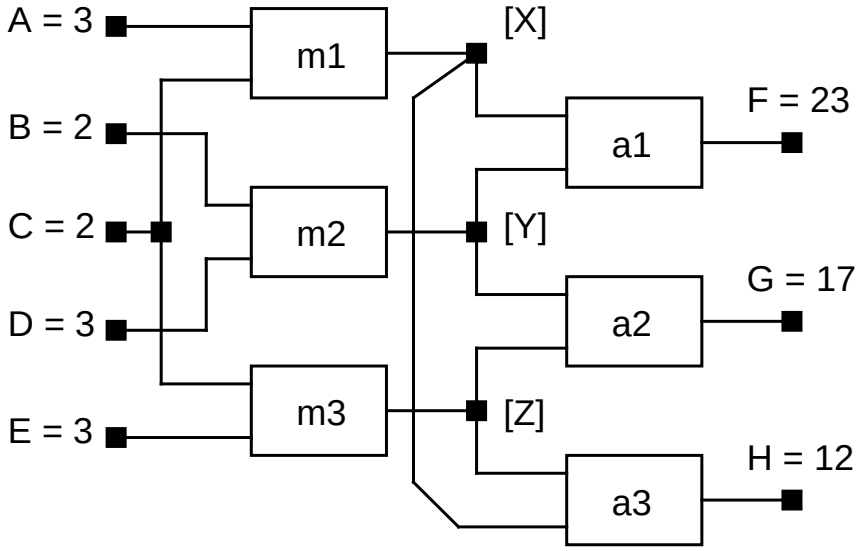

Figure 7.6: An modified arithmetic unit

Now, let us consider a very simply hierarchical model where each hierarchical level includes the same number of subcomponents k. Each path from level 0 to the last elementary component has the same length l, i.e. number of hierarchical levels is the same for each path. Conflicts between observations and calculated values can occur only at the component outputs.

The model parameters are:

- l - a number of hierarchical levels,
- k - a number of components of each level,
- x - a number of outputs from each component,
- b - a number of observed inconsistencies for one component (a number of component outputs),

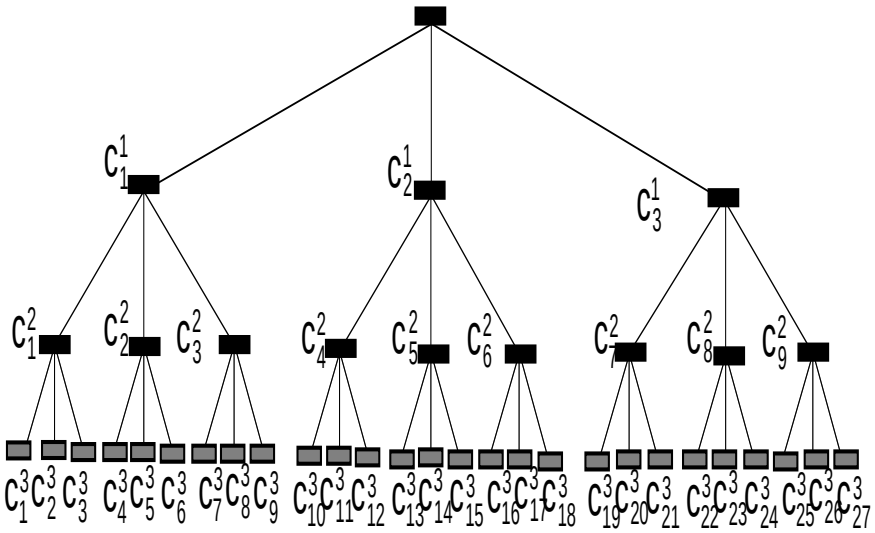

Figure 7.7: A example of hierarchical model for estimating of computational average cost of the hierarchical algorithm

- Δ_b - a number of possible different sets of values for each subcomponent.

$$\Delta_b = v_s^{max} = \sum_{i=1}^{b} \binom{b}{i}, 0 < b \leq kx.$$

A single–level model of the same system would include $n = k^l$ components.

The refutation procedure for one level with k subcomponents has to be repeated p times, where p is:

$$p \ = \ 1 \ + \ k\Delta_b \ + \ \ldots \ + \ k^{l-1}\Delta_b^{l-1} \ = \ \sum_{i=0}^{l-1}(k\Delta_b)^i.$$

The quickest growing part of the above equation is

$$k^{l-1}\Delta_b^{l-1}.$$

The quickest growing part of Δ_b is

$$\binom{b}{\lfloor \frac{b}{2} \rfloor} = \frac{b!}{\frac{b}{2}!\frac{b}{2}!}.$$

For further consideration, we can assume that it is proportional to $b!$; $\lfloor \ \rfloor$ means the floor function. Then we have that

$$p \sim k^{l-1}b!^{l-1} = (kb!)^{l-1}.$$

Now, we consider different cases of computational complexity for refutation procedure:

1. Polynomial complexity - O(n^p). For one level of the hierarchy, it is $(kx)^p$; for a non–hierarchical model of the example system, it is $(kx)^{pl}$. Then

$$(kx)^p(kb!)^{l-1} \sim (kx)^{pl},$$

$$(kb!)^{l-1} \sim (kx)^{p(l-1)},$$

$$(kb!) \sim (kx)^p,$$

$$b! \sim k^{p-1}x^p.$$

   We can consider the following cases:

   (a) $b = x = c < k$, a number of possible faulty observations has a pre–defined maximum c and it is constant value and is not a function of the number of subcomponents at a hierarchical level. Such situation can happen only when the number of component outputs is limited to the same value, then we can write $x = c$. Then

$$c! \sim k^{p-1}c^p,$$

   and the complexity for this case is lower, in most cases, than complexity for non–hierarchical description.

   (b) $b = x = k$, a number of possible faulty observations is equal to the number of subcomponents at a hierarchical level. Such situation can happen only when the number of component outputs is limited to the same value, then we can write $x = k$. Then

$$k! \sim k^{2p-1},$$

   and the complexity for this case is higher, in most cases, than complexity for non–hierarchical description.

2. Exponential complexity - O(p^n). It is p^{kx} for one level of the hierarchy, for a non–hierarchical model of the example system it is $p^{(kx)^l}$. Then

$$p^{kx}(kb!)^{l-1} \sim p^{(kx)^l}.$$

   Similarly to polynomial complexity here are the following cases:

(a) $b = x = c < k$, then

$$p^{kc}k^{l-1}(c!)^{l-1} \sim p^{(kc)^l},$$

the fastest growing part on the left side is p^{kc}, on the right side it is $p^{(kc)^l}$. The complexity for this case is lower than complexity for non–hierarchical description.

(b) $b = x = k$, then

$$p^{kk}(kk!)^{l-1} \sim p^{k^{2l}},$$

and the situation is strongly depend on a number of levels and components at levels.

3. Factorial complexity - O(n!). It is $(kx)!$ for one level of the hierarchy, for a non–hierarchical model of the example system it is $((kx)^l)!$. Then

$$(kx)!(kb!)^{l-1} \sim ((kx)^l)!.$$

(a) $b = x = c < k$, then

$$(kc)!(kc!)^{l-1} \sim ((kc)^l)!,$$

the fastest growing part on the left side is $(kc)!$, on the right side it is $((kc)^l)!$. Complexity for this case is lower than complexity for non–hierarchical description.

(b) $b = x = k$, then

$$(k^2)!(kk!)^{l-1} \sim (k^{2l})!,$$

complexity for this case is lower than the complexity for non–hierarchical description.

Roughly speaking, the limitation of complexity by hierarchization is possible for all types of refutation procedures, if the number of outputs of components is limited and lower than the number of components at a diagnosed level. Otherwise, the situation is depend on complexity of the original refutation procedure.

Chapter 8

Diagnostic process

The diagnostic process for hierarchical complex systems is similar to a typical diagnostic process and includes the following three basic steps:

- fault detection,
- generation of diagnoses (fault isolation),
- verification of diagnoses, (with some elimination procedure, additional measurement, tests etc.).

Each stage is shortly presented in the following sections.

8.1 Fault detection

The diagnostic system is working simultaneously with a diagnosed complex system. Some process variables from the diagnosed system are read by sensors and sent to the diagnostic system. They become observations OBS.

The observations are compared with either calculated behavior, for model–based components, or expected behavior, for graph–described components. If character of the observations is discrete, what mainly happens when digital units are diagnosed, then diagnostic process is easier and based on comparison between predicted values of model variables and their observed values.

A worse case is for continuous variables. They are more common but also more difficult for fault detection. It is because of measurement noise and temporary process fluctuations what change observations and make them not so easy to compare. In such a case, acceptable ranges of errors are defined and fault is detected when errors are outside of these ranges (Figure 8.1). Frequently, additional time or amplitude conditions have to be fulfilled.

Faults can be detected at each level of the hierarchical model, where a number of observations is sufficient for such detection. The sufficient number, for model–described components, has to allow to generate redundant values for some model variables: calculated ones and observed ones. For graph–described components, the number and values of observations have to allow to infer that the component behavior is either faulty or correct.

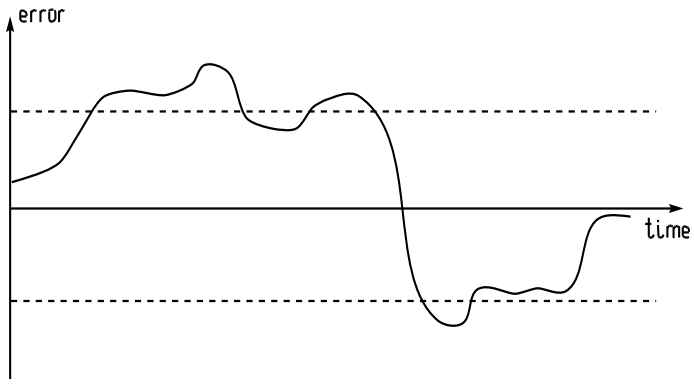

Figure 8.1: Simple example of error of a continuous observation and its upper and lower ranges

8.2 Fault isolation

If a fault occurs then all components which are antecedents of the faulty component c_i^j can be detected as faulty. Theoretically, all components c_k^l that $c_k^l \in ANTECEDENTS(c_i^j) \cup c_i^j$ can be faulty and their observations should be not consistent with their correct behavior.

If set $FAULTS$ includes all faulty components and these misbehaviors are detected for some components $c_m^n \in DETECTED$ then diagnosis is started for root component c_i^j such that

$$c_i^j \in DETECTED \ \ and \ \ ANTECEDENTS(c_i^j) \notin DETECTED,$$

what allows to find the most complete set of diagnoses. Detection is possible only in levels where a number of observations is sufficient. Further diagnosis of component c_i^j is carried out by the global diagnostic procedure. If no diagnosis is found then the procedure is repeated for next one root component which is found for new set of detected faulty components:

$$DETECTED = DETECTED - \{c_i^j\}.$$

Example 8.1 *Figure 8.2 presents a hypothetical hierarchical model where component c_3^3 is faulty and two components are detected as faulty: c_2^1 and c_5^2. Diagnosis is started for the higher component: c_2^1.*

8.3 Verification of diagnoses

There are two types of verification for the hierarchical diagnosis:

- Theoretical - based on existing observations and properties of models,

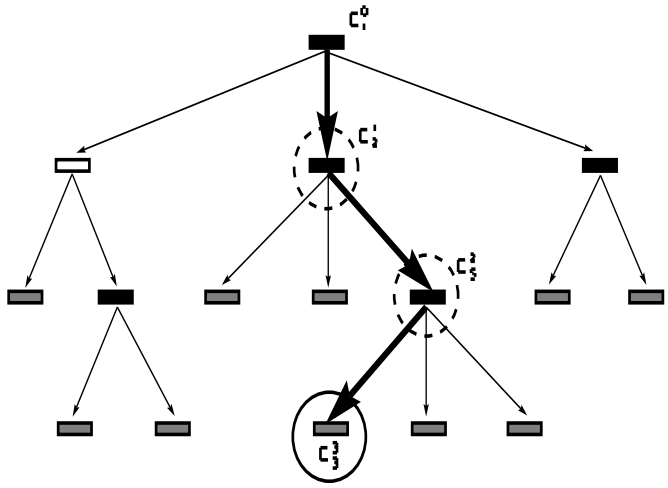

Figure 8.2: Fault detection in hierarchical model

- Physical - based on additional observations or tests which are performed after fault detection and give additional observation.

8.3.1 Theoretical verification

As it was mentioned in the previous chapter, the global diagnostic algorithm can reject some diagnoses in a few cases. Two of them are important for diagnostic verification:

- Values of attributes are out of their domains (physical impossibility),
- Diagnostic procedure at a hierarchical level is finished without any diagnosis.

Moreover, we can add here one more condition:

- Observations $OBS_i^{j/calc}$ calculated by the mapping functions F and H are inconsistent with real world observations OBS_i^j.

The first case is obvious.

Example 8.2 *For example, one from the diagnoses for the arithmetic unit in Figure 4.1 is* $\{m1\}$. *The value at the output of the faulty subcomponent* $m1$ *should be equal* 17 *for observations which are shown in the figure. We calculate it on the basis of components:* $m2$, $a1$. *Multiplier {m1} on a lower hierarchical level can looks like this one in Figure 8.3.*

The model on the lower level shows that the multiplier is not capable of producing output value equal 17 (maximum value is 15) and 17 is out of the domain of the multiplier output. Hence, diagnosis $\{m1\}$ *will be rejected because of physical impossibility.*

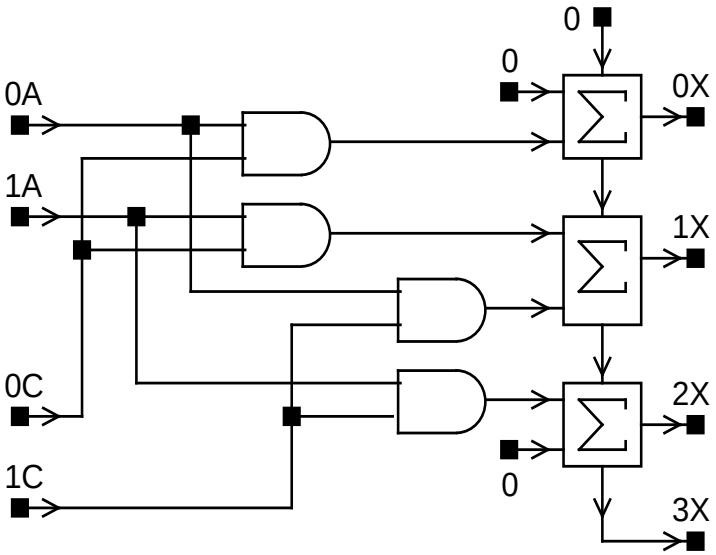

Figure 8.3: Multiplier

Assume that a component is described by set of attributes

$$ATR = ATR_d \cup ATR_u,$$

where:

- ATR_d are attributes with defined value,
- ATR_u are attributes with undefined value.

Now, if the component is refused as a part of a diagnosis then the component will be refused also for set of attributes such

$$ATR' = ATR'_d \cup ATR'_u$$

where:

$$ATR'_d \supseteq ATR_d.$$

A proof of this fact is simple for the refusing based on domain limitations for model–based components. The refusing based on conflicts in causal AND/OR/NOT graphs fulfills also the above property. If it is impossible to explain behavior of the component by the observed symptoms ATR_d and any value of undefined symptoms ATR_u, then the behavior will be also unexplained on the basis of superset of the observed symptoms ATR'_d.

The second case occurs when a component is distinguished as faulty at level j but its behavior is classificated as normal or undefined at lower level $j+1$. The reasons for such situation can be as follows:

- weakness of component description at level $j+1$,

- intentional elimination of some impossible cases at level $j+1$,
- intentional relaxation of the component description at level j.

For example, a logical function, which is represented by an causal graph, cannot be fulfilled for current values of observed symptoms and any values of undefined symptoms.

Example 8.3 *Figure 8.4 presents simple expert graph modeling gate EX–OR, the gate has two inputs* in1(X1) = 0, in2(X1) = 0 *and one output* m1(X1) = 1.

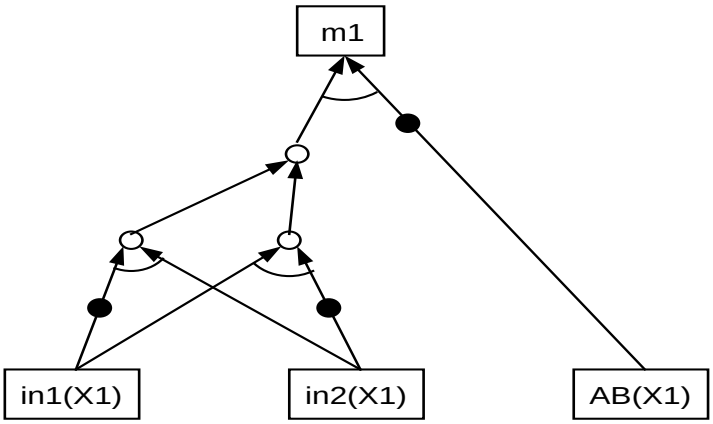

Figure 8.4: Graph AND/OR/NOT for XOR gate

The logical function coupled to the graph is:

$$m1(X) \;=\; (in1(X)\; XOR\, in2(X))\; AND\; NOT\; AB(X).$$

The above function has three observed symptoms: in1(X1), in2(X1), m1(X1)*; and one elementary diagnosis* AB(X1)*. Before mentioned values of symptoms generate conflict in the function which cannot be explained by any value of diagnosis* $AB(X)$.

Inconsistencies between observations $OBS_i^{j/calc}$ calculated by mapping functions (F and H) and real word observations OBS_i^j lead also to refusing the considered diagnosis. Differences should be significant, similarly as during fault detection. If the both observations are closed then the real word observation is taken to further calculations. Otherwise, diagnostic algorithm is stopped and returned back to supercomponent, the current examined diagnosis is skipped.

There is one more chance to theoretical verification of some diagnoses. If the same component c is reused at a few levels (Figure 8.5) then its diagnoses from the levels, sets of faulty subcomponents, can be compared.

Reusing is understand here as use of the same component by a few supercomponents, but without simultaneous influences on the component. If influence is simultaneous then the influence should be modeled at a higher hierarchical level which is common for all influenting supercomponents (Figure 8.6).

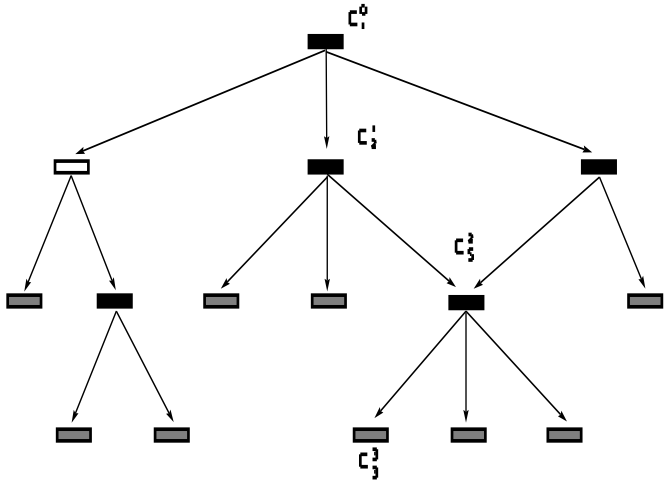

Figure 8.5: An example of reusing of component c_5^3

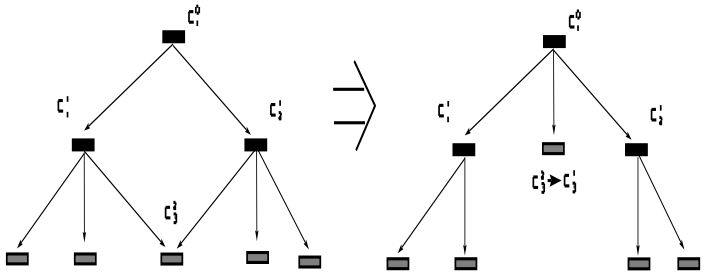

Figure 8.6: Necessary transformation for components with more than one supercomponent

A good example of correct reusing is an adder which is used by a few other electronic units. The simplest strategy for such comparision of diagnoses of the subcomponent is that only these diagnoses which are repeated will be further considered at lower hierarchical levels.

8.3.2 Verification based on physical tests

Tests can be performed during both the local and the global diagnostic procedure. Strategy of verification for the local diagnostic procedure depends on type of component description. For example, a simple and efficient methodology for verifyng diagnoses for AND/OR/NOT graphs is described in (Ligęza, 2003).

When we consider verification in the global diagnostic algorithm then the following conditions seem to be rational:

- a number of tests should be as less as possible, tests are selective i.e. number of possible diagnoses should be maximally reduced by a single test,
- tests should be as easy as possible to perform, i.e. the test should be easy to perform.

Assume that after a test on subcomponent c we know that subcomponent c is faulty or not. $D = \{d_1, d_2, \ldots, d_l\}$ is a collection of diagnoses. Then a number of diagnoses which include subcomponent c is defined as follows

$$n(c) = |\{d_i\}| : d_i \in D \wedge c \in d_i.$$

Now, let us define a cost function m as a simple weight function such that:

$$m(c) = w_n n(c) + w_p p(c) + w_r r(c),$$

where

- w_n is a weight for the number n of diagnoses including subcomponent c,
- w_p is a weight for the a priori probability p of malfunction of subcomponent c (probability of malfunction is discussed in the next section); this value is usually below zero (we want to refuse the diagnoses),
- w_r is a weight for the function r describing how easy is to check that subcomponent c is either faulty or not.

The weights can be set for each component either empirically or a priori. Subcomponents from single–element diagnoses can be also preferred.

The best candidate to testing is the subcomponent for which the cost function has the highest value, what can be written as

$$b(c^*) = \max_{c \in d_1, d_2, \ldots, d_l} (m(c)).$$

The global diagnostic procedure diagnoses sometimes the same component a few times with different sets of values of attributes (input/output values). It is also possible that a component is reused (not simultaneously). Then a collection of diagnoses D_j is produced as a result of each diagnosis. Now, the verification can be done after finishing the global diagnostic procedure (based on depth–first search). Now, the number of diagnoses including subcomponent c is formulated as follows:

$$n_G(c) = |\{d_i\}| : (d_i \in D_1 \vee d_i \in D_2 \vee \ldots \vee d_i \in D_p) \wedge c \in d_i.$$

and

$$m_G(c) = w_n n_G(c) + w_p p(c) + w_r r(c).$$

As before, the best candidate to testing is the subcomponent for which the new cost function has the highest value.

8.4 Probabilistic information in the hierarchical diagnosis

Diagnoses generated for a complex system can be ordered according to probability of particular diagnoses. Such ordering allows to improve diagnostic process. More likely diagnoses are tested first and probability that real causes of system malfunction will be found at the beginning is higher. So, estimated time of reparation is shorter and efficiency of diagnosis is increased.

The next section describes approaches to calculating probability of diagnoses for model–described and graph–described *complex components*. After that, a methodology for calculation probabilities for all system diagnoses is presented.

8.4.1 Probability in the local diagnostic algorithm

A diagnosis d_k of a complex component c_i^j is defined as a set of subcomponents $\{c_{k1}, \ldots, c_{kn_k}\}$ that all have to be faulty for correct explanation of observations. The diagnoses can be written as

$$AB(c_{k1}) \wedge \ldots \wedge AB(c_{kn_k})$$

If each component in the diagnosis has a priori assigned probability of malfunction $P_m(c_i) = q_i$ then the simplest way to assign probability to the whole diagnosis is choice a minimum from these probabilities

$$P_d(d_k) = min(P_m(c_{k1}), \ldots, P_m(c_{kn_k})).$$

Ordering of probabilities q_i is done according to *order relation* $\succeq$ that it is possible to appoint such minimum value even when they are linguistic symbols e.g. „less probably", „more probably". A minimum norm for conjuncted probabilities is popular strategy in many works, for example: French (1988) - AI, Zadeh (1965) - fuzzy logic.

Probabilities of diagnoses can be also calculated according to a priori probabilities and the component description. The most popular approach to such calculations is based on conditional probability. Below short survey presents a few methodologies where probability is used to improve a diagnostic process.

Causal AND/OR/NOT graphs

Calculation of probability of diagnoses for graph–described components is based on abductive analysis of causal structures using „qualitative probabilities" French (1988), Fuster-Parra and Ligęza (1995).

Let us define *qualitative probabilities*:

- $\succeq$ denote a week order relations, greater event is more probably,
- Q denote a set of ordered elements $Q = \{q_1, q_2, \ldots, q_n\}$ that $q_n \succeq q_{n-1} \succeq \ldots \succeq q_2 \succeq q_1$,
- $\overline{Q}$ denote a set of ordered elements $\overline{Q} = \{\overline{q}_1, \overline{q}_2, \ldots, \overline{q}_n\}$ that $\overline{q}_1 \succeq \overline{q}_2 \succeq \ldots \succeq \overline{q}_{n-1} \succeq \overline{q}_n$,
- $q_1 \succeq \overline{q}_1$.

Generally, the probabilities are linguistic statements (i.e. expert–provided) concerning the likelihood of certain events e (i.e. $q_i = very\ likely$). Qualitative probabilities $\overline{q}$ are associated to the events e' which are complementary to e.

Definition 8.1 (Fuster–Parra) *Two events e and e' are called complementary events when:*

1. *either e or e' has to occur,*
2. *it is not possible that e and e' occur at the same time,*
3. *$\{e\} \cup \{e'\}$ = complete set of possibilities.*

Let $G = (N, E_*, E_-)$ be an AND/OR/NOT causal graph. Qualitative a priori probabilities of fault occurrence for elementary diagnoses D are known, they are elements of $Q \cup \overline{Q}$. Now, we define the way of assigning qualitative probabilities to nodes in the graph in the following way:

Definition 8.2 (Fuster–Parra) *Let $Q \cup \overline{Q}$ denote the set of qualitative values under consideration, and let $N = D \cup V \cup M$ be the set of graph nodes (D - a set of elementary diagnoses, V - a set of pre–specified intermediate symptoms, M - a set of manifestation symptoms).*

A qualitative probability assignment function λ is a function such that:

- *λ assigns qualitative probabilities directly to elementary faults, i.e.*

$$\lambda : D \longrightarrow Q \cup \overline{Q}$$

  in a consistent way (elements of D are mapped into elements of $Q \cup \overline{Q}$),

- *in the case of an OR node: let $(n_1, n_i), \ldots, (n_{i-1}, n_i) \in E^2$, and let q_j denote the qualitative probability assigned to $n_j, j = 1, 2, \ldots, i-1$; then the qualitative probability q_i of n_i is calculated as $q_i = \bigvee(s_1, s_2, \ldots, s_{i-1})$,*

- *in the case of an AND node: let* $(n_1, n_2, \ldots, n_{i-1}, n_i) \in E^i$*, and let* q_j *denote the qualitative probability assigned to* $n_j, j = 1, 2, \ldots, i-1$*; then the qualitative probability* q_i *of* n_i *is determined as* $q_i = \bigwedge(s_1, s_2, \ldots, s_{i-1})$*„*
- *in the case of a NOT arc: let* $(n_1, n_2) \in E^-$*, and let* q_1 *denote the qualitative probability assigned to* n_1*; the qualitative probability* q_2 *of* n_2 *is calculated as* $s_2 = \overline{q}_1$.

When we know diagnoses then it is possible to propagate qualitative probabilities, which are assigned to elementary diagnoses, up to manifestations. In this way, each diagnosis will have probability.

Example 8.4 *An example expert graph is shown in Figure 8.7. Labels besides intermediate symptoms describe the propagation rules using for calculation of symptom probabilities.*

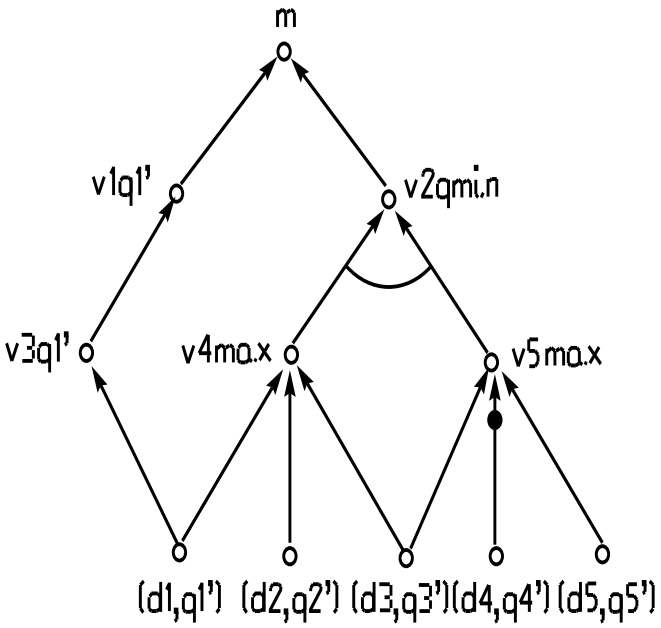

Figure 8.7: A casual graph for diagnosing

If qualitative probability is not used then sets of potential diagnoses are given by:

1. $D_1 = \{d_1/true\}$
2. $D_2 = \{d_1/true, d_3/true\}$
3. $D_3 = \{d_1/true, d_4/false\}$
4. $D_4 = \{d_1/true, d_5/true\}$

5. $D_5 = \{d_2/true, d_3/true\}$
6. $D_6 = \{d_2/true, d_5/true\}$
7. $D_7 = \{d_2/true, d_4/false\}$
8. $D_8 = \{d_3/true\}$
9. $D_9 = \{d_3/true, d_5/true\}$
10. $D_{10} = \{d_3/true, d_4/false\}$

Minimal diagnoses with respect to set inclusion are as follows:

1. $D_1 = \{d_1/true\}$
2. $D_8 = \{d_3/true\}$
3. $D_6 = \{d_2/true, d_5/true\}$
4. $D_7 = \{d_2/true, d_4/false\}$

When we assume that an order of qualitative probabilities of elementary diagnoses $d_1, d_2 \ldots, d_5$ is as follows $q'_5 \succeq q'_4 \succeq \ldots \succeq q'_1$ then qualitative probabilities assigned to specific nodes are: $v_3 = q'_1$, $v_4 = q'_3$, $v_5 = q'_5$, $v_2 = q'_3$, $v_1 = q'_1$. The most probably diagnosis generated by the probabilistic version of search procedure is diagnosis $D_9 = \{d_3/true, d_5/true\}$.

Models

Most model–based approaches, where probability is taken into consideration, are based on Bayes rule. The result is usually probability of malfunction calculated for each component when observations have specified values.

Notion of a „candidate" is proposed by De Kleer and Wiliams (de Kleer and Williams, 1987). A candidate is a set of components that a malfunction of the components make the observation compatible with the system description. Rest of components, which do not belong to the candidate, is considered as working correctly.

The prior probability that a particular candidate C_l is the current one is:

$$p(C_l) = \prod_{m \in C_l} p(m).$$

where $p(m)$ is the prior probability of behavior mode m being manifested i.e., a particular component being in a particular mode. Now, some candidates can be eliminated by additional measurements (e.g. by test when variable x_i is measured with value v_{ik}), the probabilities of the rest of diagnoses are increased and can be calculated by the Bayes rule. Let $x_i = v_{ik}$ that v_{ik} is a measured value of variable x_i.

$$p(C_l|x_i = v_{ik}) = \frac{p(x_i = v_{ik}|C_l)p(C_l)}{p(x_i = v_{ik})}.$$

where:
$p(C_l)$ - was computed as a result of previous measurement or is the prior probability,
$p(x_i = v_{ik}|C_l)$ - is determined as follows:

- If $x_i = v_{ik}$ is predicted by C_l given the evidence so far then $p(x_i = v_{ik}|C_l) = 1$.
- If $x_i = v_{ik}$ is inconsistent with C_l and the evidence then $p(x_i = v_{ik}|C_l) = 0$.
- If $x_i = v_{ik}$ is neither predicted nor inconsistent with C_l and the evidence then we make presupposition (sometimes invalid) that every possible value for x_i is equally likely. Hence, $p(x_i = v_{ik}|C_l) = \frac{1}{m}$ where m is the number of possible values x_i might have (e.g. in a conventional digital circuit $m = 2$).

$$p(x_i = v_{ik}) = p(S_{ik}) + \frac{p(U_i)}{m}.$$

where:
S_{ik} - is a set of candidates that $x_i = v_{ik}$ is predicted by them. U_i - is a set of candidates that value of x_i is undefined.

De Kleer and Williams also show how minimal entropy techniques can be used to guide the selection of the variable to be observed next. This methodology needs conditional probability of some observed variables (candidates) what may be difficult in some cases.

Pearl uses Bayesian networks for modeling probability of malfunctions of components (Pearl, 1991). The model is transformed into causal graph and after that, one more time, into Bayesian network. Probabilities of input nodes and malfunction of the components are given a priori, other nodes have defined link probabilities. When some observations on outputs are known, the posterior probabilities of the nodes corresponding to components of the circuit are computed using the traditional message propagation algorithm. The result is a mathematical formula giving the posterior probability that a component is faulty.

8.4.2 Probabilistic information in the global diagnostic algorithm

The global diagnostic algorithm and the test procedure can be improved by use of probabilistic information.

The global diagnostic procedure (depth–first search) can implement the greedy strategy. Collection of diagnoses $D = \{d_1, d_2, \ldots, d_{k_D}\}$, which is generated by the local diagnostic algorithm for a complex component c_i^j, is analyzed in order to probability of diagnoses $P = \{p_1, p_2, \ldots, p_{k_D}\}$. The order is from the most probably

$$d_{max} = d_k : d_k \in D \wedge P_d(d_k) = p_{max}$$

to the least probably one:

$$d_{min} = d_k : d_k \in D \wedge P_d(d_k) = p_{min}.$$

The most probably diagnosis d_{max} is analyzed first and its subcomponents are diagnosed on lower hierarchical levels. Such an approach gives possibility to stop the global diagnostic procedure after generation of a few most probably diagnoses. When the global

diagnostic algorithm finishes analyzing diagnosis d_{max} at lower hierarchical levels (depth first strategy) then collection is modified to $D = D - \{d_{max}\}$ and next maximal probable diagnosis can be taken.

The test procedure can be improved in a similar, simple way. Assume that probability q_i of malfunction for each subcomponent c_i in diagnosis d_n is known. The subcomponents from diagnosis d_n should be tested in order to probability of their malfunction q_i, from the least probably subcomponent

$$c_{min} = c_k : c_k \in d_n \wedge P_m(c_k) = q_{min}$$

to the most probably subcomponent

$$c_{max} = c_k : c_k \in d_n \wedge P_m(c_k) = q_{max}.$$

The order allows to refuse the diagnosis in a minimal number of tests because all subcomponents in a diagnosis have to be faulty. Therefore the best way to refuse the diagnosis is to test firstly subcomponents with low possibility of malfunctions.

Part IV

Examples and conclusions

Chapter 9

Examples

9.1 Graphical representation of diagnoses

An interaction between a human and the diagnostic system requires a friendly form of presentation of diagnoses, which should be also capable to explain how diagnoses are generated. The proposed graphical form is based on the intermediate graphs and can be easily implemented as a computer application.

Graphical representation of results obtained from the global diagnostic procedure is a combination of intermediate graphs obtained from the local diagnostic procedure.

Example 9.1 *Figure 9.1 presents in graphical form results of the global diagnostic procedure for a hierarchical model of the arithmetic system in Figure 4.1.*

Elementary components are marked as empty squares (e.g. $a1$*). Squares with a cross mean that component has diagnoses and the subcomponents in diagnoses can be displayed (e.g.* $m1$*). If subcomponents of a component are currently displayed then there is a minus mark inside a square (e.g.* $m2$*). A square is gray if a component is described by model (e.g.* $a2$*). A square is green if a component is described by graph.*

There are three multipliers and two adders at the highest level of the model. Conflict sets for the observations are: $\{a1,\ m1,\ m2\}$, $\{a1, a2, m1, m3\}$. *Minimal hitting sets for the observations are:* $\{a1\}$, $\{m1\}$, $\{a2, m2\}$, $\{m2, m3\}$. *Components* $a2, m1, m2, m3$ *include subcomponents.*

Diagnoses for a graph–described component can be presented directly as an intermediate graph connected to a node which is a representation of the component.

A different situation is with model–described components. There are two solutions of this problem. The first one is trivial; graphs for model–described components should be similar to these ones for graph–described components and include all possible diagnoses (e.g. minimal hitting sets). The second solution is that only one diagnosis can be chosen from the intermediate graph and only this diagnosis will be graphically "expanded" on lower hierarchical levels.

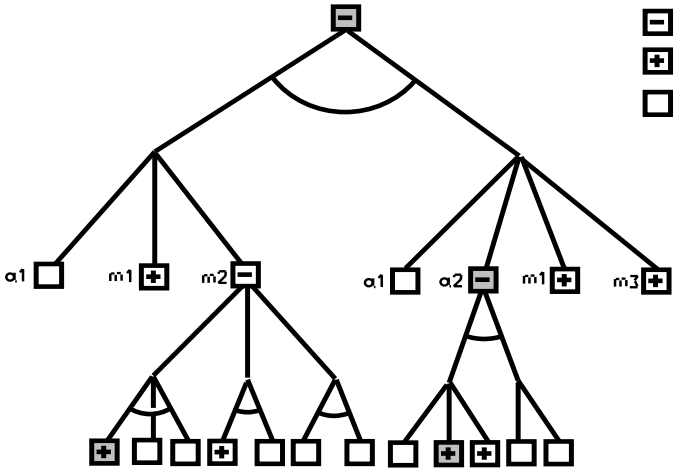

Figure 9.1: Graphical presentation of results

9.2 Conceptual system for hierarchical modeling

The proposed conceptual system is a computer application with graphical interface, its main aim is to support design of hierarchical models of complex systems. The Diagnosis Support System (DSS) can be also used for monitoring and control of the diagnostic process.

Diagnosed systems can be modeled and presented with a few basic views:

- *Level view* – showing a diagnostic description of a specified level. This view gives possibility to input current values of observations. There are two basic kinds of this view:
  - *AND/OR/NOT view* – visualization of graph–described levels,
  - *Model view* – visualization of model–described levels.
- *Hierarchical model view* – presentation of all components together with marked dependencies among them. This view gives easy access to *level views*.
- *Diagnostic view* – showing calculated diagnoses in graphical form which is presented in the previous section. This view gives access to *level views* and can be used to control diagnostic reasoning.

Hierarchical model views and *level views* are most important during development of diagnostic models. The user can define hierarchical structure of a complex system at *hierarchical model view*; the type of description of components can be also defined here.

Now, each component in the hierarchical model can be “open” and shown in the separate *level view*. All its subcomponents, which were defined at *hierarchical model view*,

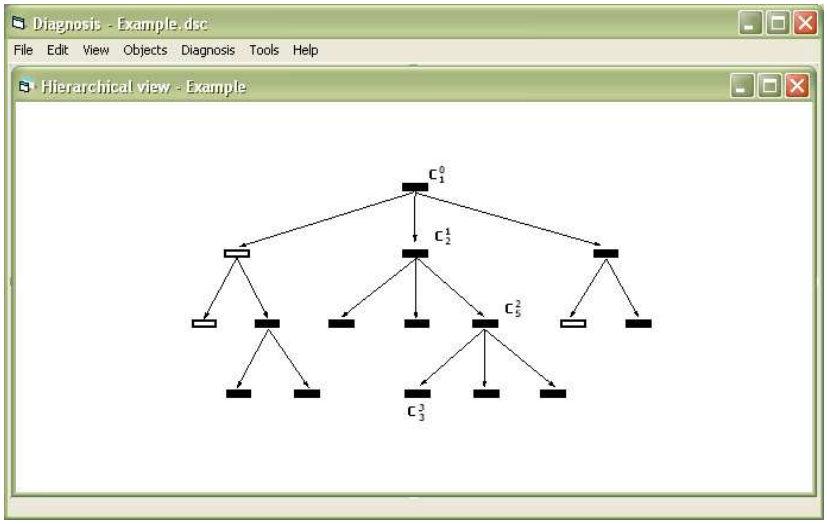

Figure 9.2: Hierarchical model view

will be accessible. Additionally, the user can add some *elementary components* and elements which are specific for a type of description e.g. connections among components for model–described components or arcs and intermediate nodes for graph–described components.

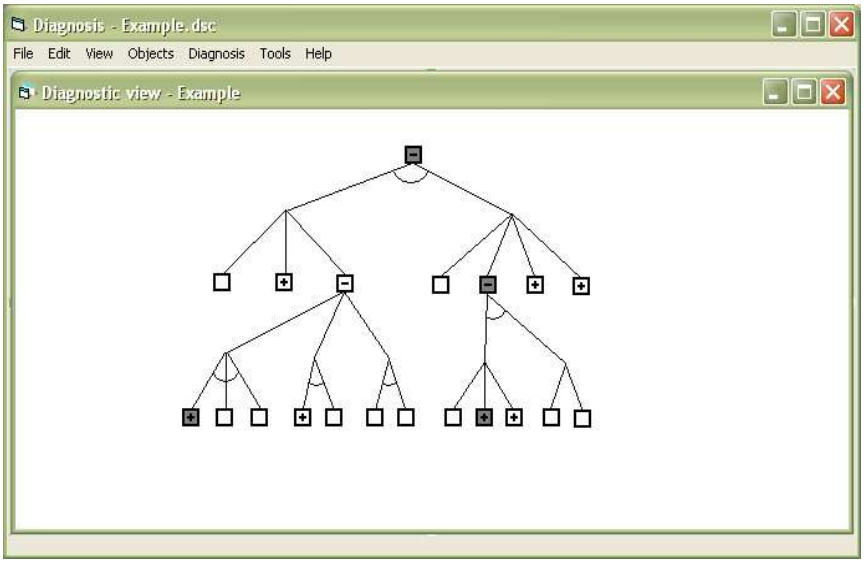

Figure 9.3: Diagnostic view

When developing is finished, it is possible to input values of some system variables i.e. input observations. Now, the user can choose a direction of diagnosis at lower levels

by selecting a diagnosis from already diagnosed levels; the form of *diagnostic view* is presented in Figure 9.3.

9.3 Example I

The next considered system is a part of a larger filling system. The aim of the system consists in keeping a constant level of liquid in tanks T_I and T_{II}. The tank set (Figure 9.4) includes:

- T_M – main tank,
- T_I – outlet tank I,
- T_{II} – outlet tank II,
- V_M – main controlled shut-off valve,
- V_I – control valve for outlet tank I,
- V_{II} – control valve for outlet tank II,
- V_{MI} – controlled shut-off valve between tanks M and I,
- V_{MII} – controlled shut-off valve between tanks M and I,
- S_M – level sensor in main tank M,
- S_I – level sensor in main tank I,
- S_{II} – level sensor in main tank II,
- C – digital controller,
- P – power supply.

9.3.1 Diagnostic description

The model of the tank system is presented in Figure 9.5. The tanks are controlled separately. Hence, controller C can be divided into three virtual controllers C_M, C_I and C_{II}.

The model of controled valves (“Valve - I”, “Valve - II”) is shown in Figure 9.7. There are the following subcomponents and variables:

- P_S - power adapter,
- C_N - control signal,
- W_P - input water pressure,

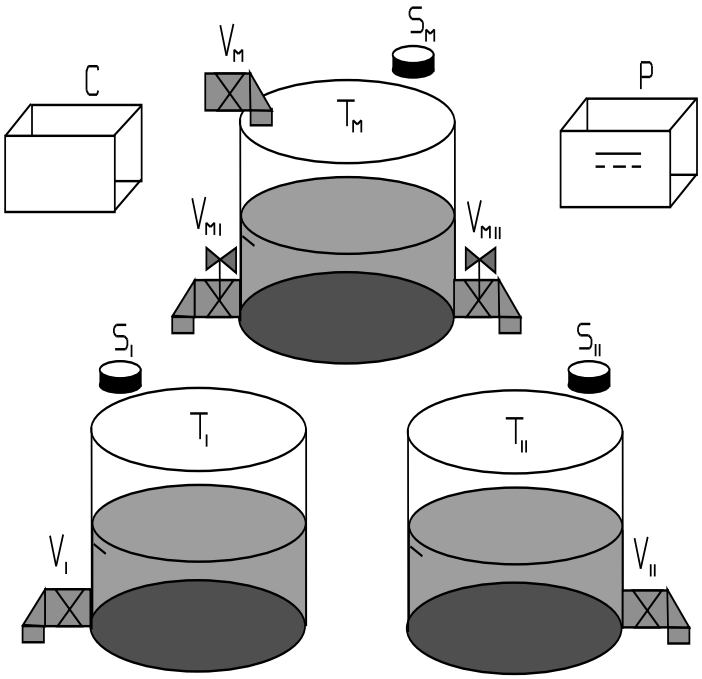

Figure 9.4: Tanks

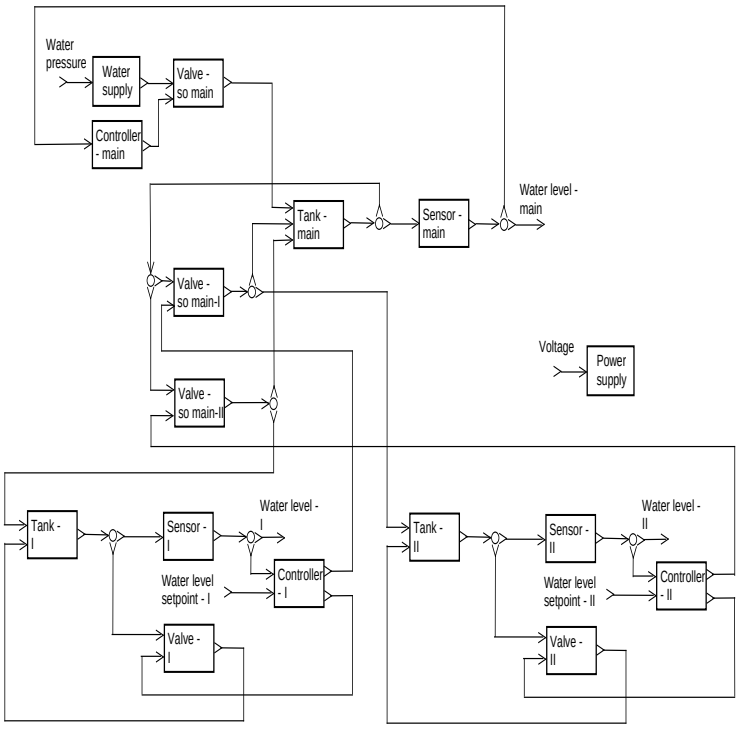

Figure 9.5: Schematic diagram of the tank system

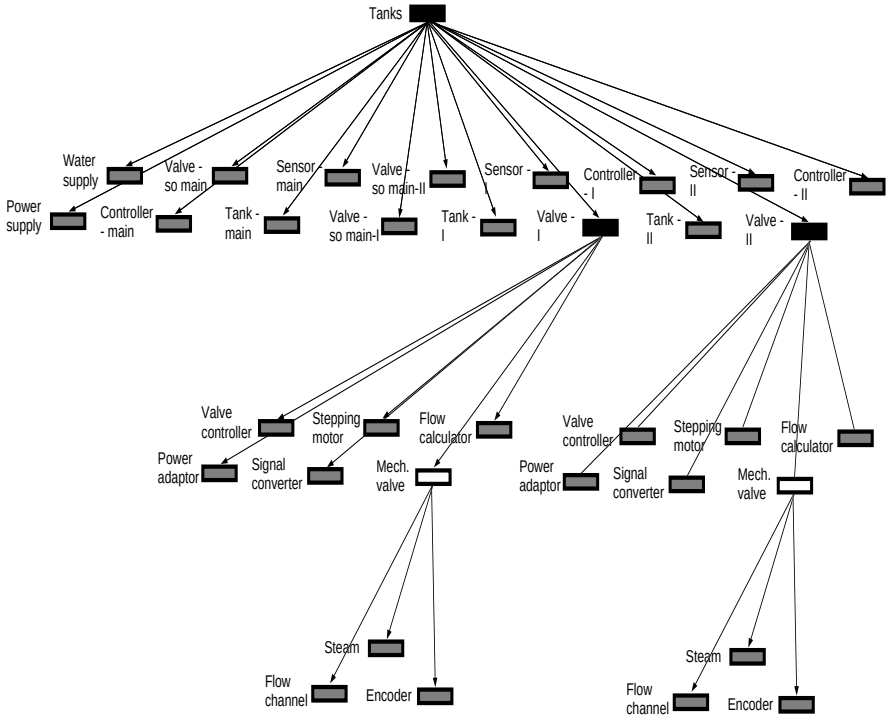

Figure 9.6: The tree of component structure for the tanks

- C_M - valve controller,
- D_A - signal converter,
- M - stepping motor, positioner,
- V_M - mechanical valve,
- $*$ - virtual component recalculating valve opening to flow,
- F_L - water flow.

There is no point in a deeper diagnosis of valve controller and signal converter, because they are integrated in one circuit. Let us diagnose a mechanical part of the valve V_M (Figure 9.8).

Diagnosis on the basis of presented model can be carried out and diagnoses can be correctly calculated. Because of the feedback loops at level 0 and simple structure of models at lower levels, there are better results if some additional observations (besides inputs and outputs of components) are available (e.g. position of valves).

9.4 Example II

Let us consider a printer as a next diagnosed object. Let the printer include four basic components at the highest level of the hierarchical model:

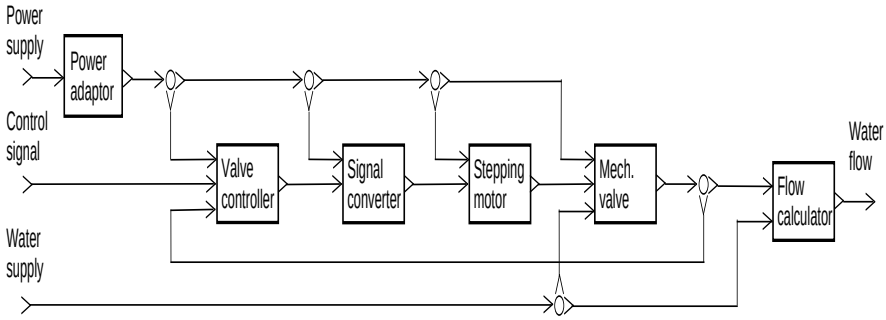

Figure 9.7: Valve schema

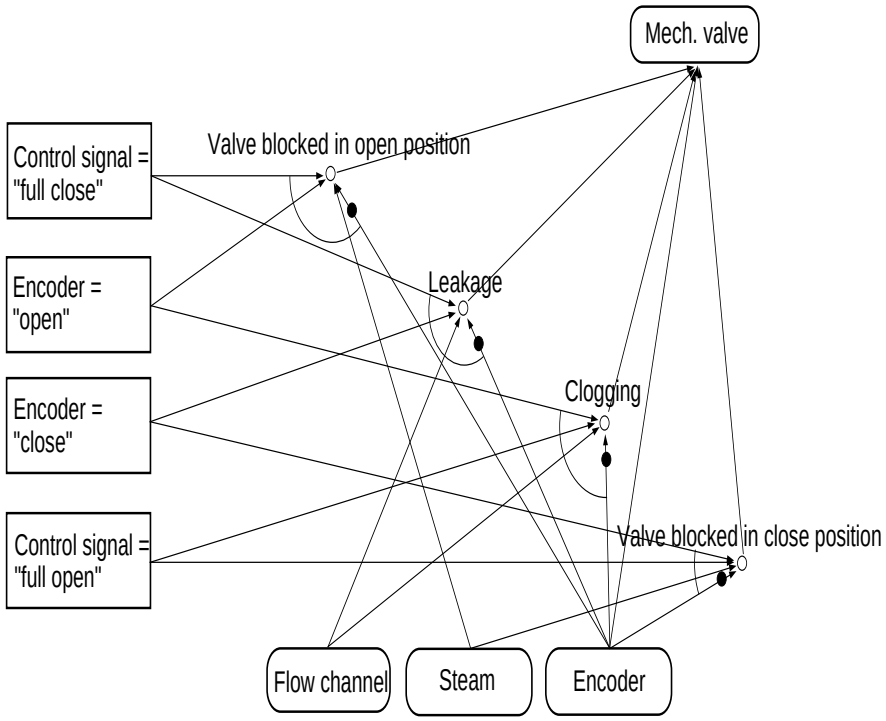

Figure 9.8: Causal graph for electromagnetic valve V_M

- feeder of paper - loading of sheets of paper to printing position and unloading after the printing,
- printing unit - printing head with ink cartridge,
- power supply - set of cords and power adaptors,
- control unit - all electronic devices responsible for printing and communication with computers.

9.4.1 Diagnostic description

The *tree of component structure* for presented printer is shown in Figure 9.9. The description is not a full technical description. It includes only these causes of malfunctions which can be either fixed by an ordinary user or interpreted as a suggestion that the printer has to be taken to a professional service.

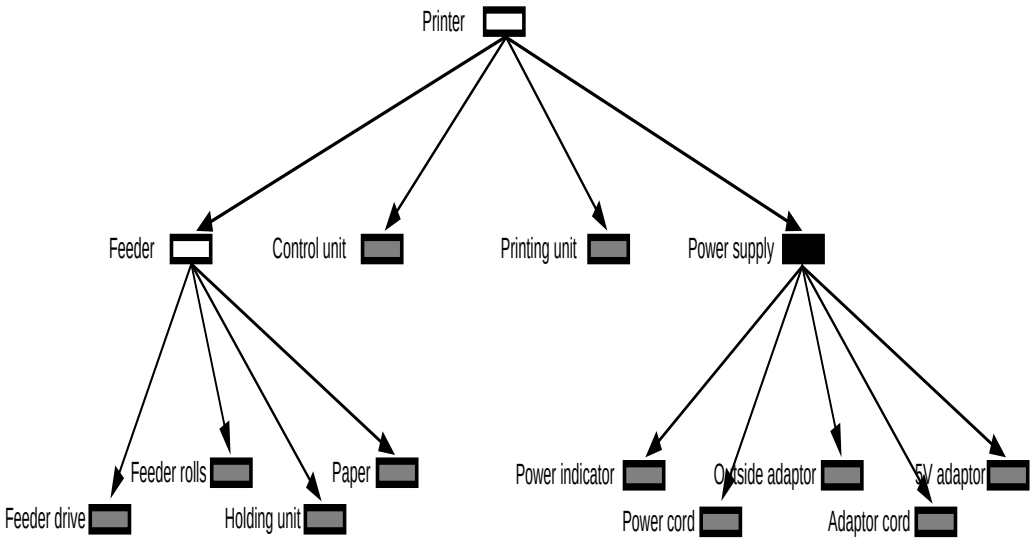

Figure 9.9: The tree of component structure for the printer

Component $Printer$ at the highest level is graph–described (Figure 9.10). Power adapter (Figure 9.11) is described below; feeder subsystem (Figure 6.1) is presented in Chapter 6. The feeder and the printing unit are described by causal AND/OR/NOT graphs, the power supply - by model.

Diagnosis on the basis of presented model can be carried out and diagnoses can be calculated. There are a few components which are described by graphs. Therefore, there are no too much mapped information and quantity of diagnoses is high. Such situation can be improve by defining values of some additional symptoms for graph–described components.

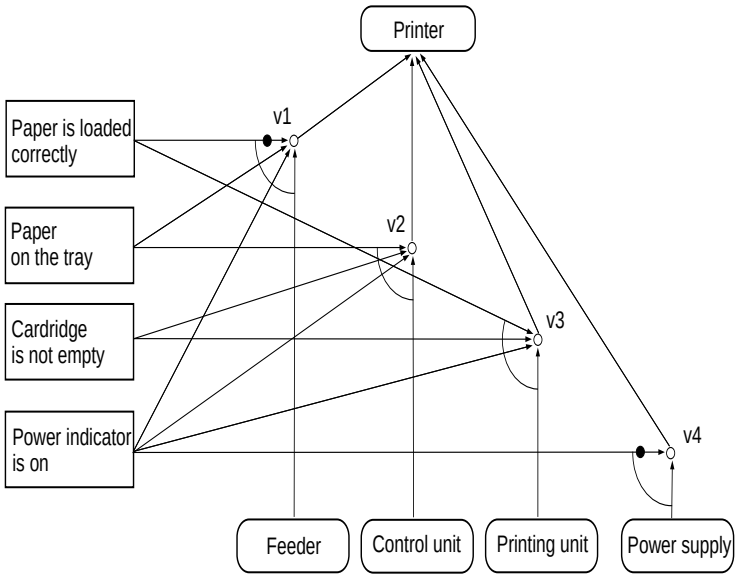

Figure 9.10: Diagnostic description for level 0 of hierarchical model of the printer

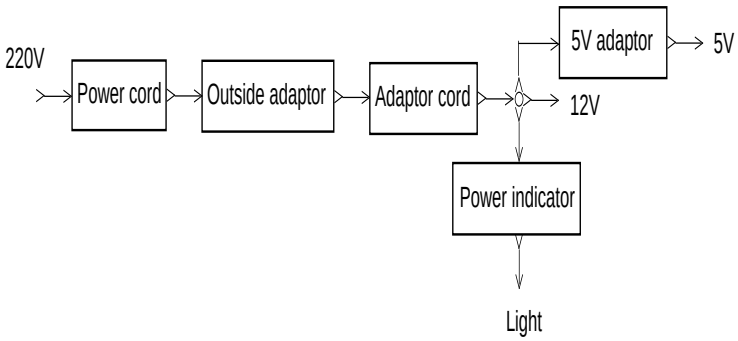

Figure 9.11: Diagnostic description for power adaptor

Chapter 10

Concluding remarks and further works

10.1 Concluding remarks

Let us briefly summarize the most important concepts presented in this thesis. We consider the diagnostic process to be a hierarchical search procedure carried out through levels of the hierarchical model. This approach allows to divide a large model on less complicated components and hierarchical levels. Such operation increases the efficiency of the diagnostic process. Moreover, we assume that components at specified levels can be described by different kinds of AI based diagnostic knowledge what gives possibility to more flexible description of technical systems.

The main results of this disertation are:

- Development of a methodology for hierarchical modeling of complex technical systems with different kinds of diagnostic descriptions for components. The definition of the generic model and elements of theory were presented. The computer application for graphical creating of such hierarchical models was described.
- Development of a methodology for hierarchical diagnostic reasoning in the above hierarchical model and, based on the methodology, a diagnostic procedure for calculating hierarchical diagnoses. Two functions responsible for inter–level symptoms mapping were generally designed, and precisely described for two kinds of component diagnostic descriptions: models and causal graphs. The computational complexity of the presented diagnostic procedure is strongly depend on models and complexity of the local diagnostic procedures, but hierarchical diagnosis is less costly than single-level diagnosis in most typical cases.

More precisely, the thesis presents the following new solutions and issues:

- the state of the art and existing AI approaches to diagnosis were presented in Chapters 1 – 4. Especially, consistency-based ones in Chapter 4 and causal ones in Chapter 3,
- definition and a formal concept of a new generic model for modeling hierarchical structures of complex systems were presented in Chapter 5 and Chapter 6,

- the methodology for hierarchical diagnostic reasoning were developed in Chapters 7 – 8,
- conceptual experimental software supporting hierarchical modeling and analysis of complex technical systems was presented in Chapter 9,
- auxiliary study of computational complexity of diagnostic reasoning in case of hierarchical approach was done in Chapter 7.

Two possible extensions have been put forward, mostly for enhancing the diagnostic efficiency. They have been as follows:

- Verification of diagnoses in the hierarchical model has been defined and verification procedure has been developed. The verification procedure uses properties of the hierarchical model and qualitative solving for selecting efficient tests,
- Search ordering with use of probabilistic information was presented. Diagnoses for levels and entire systems are ordered with respect to expert knowledge and defect statistic. The solution efficiently improves verification of diagnoses.

The thesis presented in Chapter 1 about qualitatively new and less complex diagnosis of technical systems seems to be fully confirmed by the presented theory and results.

Original results of the thesis include:

1. Development of a formal framework and an original methodology for hierarchical modeling of complex systems with heterogeneous diagnostic descriptions.
2. Design of an efficient diagnostic procedure with “focus” effect (diagnosis of subsystems as separate systems) for the hierarchical model.
3. Development of a verification procedure for the hierarchical diagnostic methodology. The procedure uses properties of the hierarchical model.
4. Development of a methodology for using expert knowledge and defect statistics for ordering diagnoses and improving efficiency of the diagnostic procedure.
5. Design of a computer application for supporting diagnostic procedure with hierarchical modeling of complex systems.

The biggest weakness of the presented approach seems to consist in the tree structure of component inclusion. There cannot be situation that two different supercomponents have simultaneous influences on the same subcomponent, i.e. the component has two supercomponents. If such situation exist, then the influences have to be modeled at a higher level which is common for both supercomponents.

All the presented ideas stay in close relations to engineering practice and can be used in diagnostic support systems what improving efficiency of search for a final diagnosis and make it more easy for users. This approach allows to describe and diagnose systems which cannot be described by a single kind of diagnostic description. Moreover, models are more readable and easier to build.

10.2 Further works

The most important further work seems consideration of different levels of precision for component attributes. Sometimes attributes cannot be calculated unambiguous, such situation occurs when some inputs of components cannot be backward calculated and no other analytical calculation is possible, e.g. value of output of a component is a comparison of two inputs. Then some levels of precision can be defined and the original approach can be extended e.g.

- *precise level* – a domain of a variable is not limited,
- *deviation level* – a domain of a variable includes only three literal symbols: *„0”* in norm, *„-1”* - below norm and *„+1”* - above norm,
- *binary level* – a domain of a variable includes two symbols: *„faulty”* and *„correct”*.
- *null level* – value is unknown.

A second direction of work is around components which can be described by different kinds of component models (than ones presented here). A lot of potential approaches can be taken from both domains: Computer Science and Control Theory.

Bibliography

Bacchus F., Yang Q., 1994. *Downward refinement and the efficiency of hierarchical problem solving*. Artificial Intelligence, vol. 71, pp. 43–100.

Bach C., Allemang D., 1996. *Case-based reasoning in diagnostic expert systems*. Artificial Intelligence Communications, vol. 9(2), pp. 49-52.

Bakker R.R., Bourseau M., 1992. *Pragmatic reasoning in model–based diagnosis*. Proc. European Conference on Artificial Intelligence ECAI'92, pp. 734–738.

Barlow R.E., Lambert H.E., 1975. *Introduction to fault tree analysis*. In: R.E. Barlow, J.B. Fussell and N.D. Singapur–Walla (Eds.). *Reliability and Fault Tree Analysis. Theoretical and Applied Aspects of System Reliability and Safety Assessment*. Society for industrial and applied mathematics (SIAM), 7–35.

Bratko I., Mozetič I., Lavrač N., 1989. *KARDIO: A Study in Deep and Qualitative Knowledge for Expert Systems*. The MIT Press, Cambridge, MA.

Brown J.S., Burton R.R., de Kleer J., 1982. *Pedagogical, natural language, and knowledge engineering techniques in SOPHIE I, II, and III*. In: Sleeman D., and Brown J.S. (eds). *Intelligent Tutoring Systems*. pp. 227–282, Academic Press, New Jork.

Bundy A., Giunchiglia F., Sebastiani R., Walsh T., 1996. *Calculating Criticalities*. Artificial Intelligence, 88(1–2).

Chen J., Patton R.J., 1999. *Robust Model Based Fault Diagnosis for Dynamic Systems*. Boston, Kluver Academic Publishers.

Chow E.Y., Willsky A.S., 1984. *Analytical redundancy and the design of robust failure detection systems*. IEEE Transaction Automatic Control, vol. 29, no. 3, pp. 603–614.

Console L., Dupre D. T., Torasso P., 1989. *A theory of diagnosis for incomplete causal models*. Proceedings IJCAI'89, 1311–1317.

Console L., Torasso P., 1992. *A spectrum of logical definitions of model-based diagnosis*. In: *Reading in Model-Based Diagnosis* (Hamscher, W., Console, L., dee Kleer, J., Eds.). Morgan Kaufmann Publishers, pp. 78–88.

Cordier M-O., Dague P., Dumas M., Lévy F., Montmain J., Staroswiecki M., Trave-Massuyes L., 2000. *A comparative analysis of AI and control theory approaches to model–based diagnosis*. In: ECAI 2000, 14th European Conference on Artificial Intelligence (W. Horn, Ed.), pp. 136–140, IOS Press, Berlin.

Dague P., Raiman O., Devés P., 1987. *Troubleshooting: When modeling is the difficulty*. Proc. American Association for Artificial Intelligence, AAI'87, Seattle, WA, pp. 600–605.

Darwiche A., 1995. *Model–based diagnosis using causal networks*. In Proc. IJCAI 95, pp. 211–217, 1995.

Darwiche A., Provan G., 1997. *The Effect of Observations on the Complexity of Model–Based Diagnosis*. Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97), pp. 99-104.

Davis R., 1984. *Diagnostic reasoning based on structure and behavior*. Artificial Intelligence, vol. 24, pp. 347–410.

de Kleer J., 1976. *Local methods for localizing faults in electronic circuits*. MA: MIT AI Memo 394.

de Kleer J., Brown J.S., 1986. *Theories of causal ordering*. Artificial Intelligence, vol. 29, pp. 33–61.

de Kleer J., 1986. *An assumption–based TMS*. Artificial Intelligence, vol. 28, pp. 127–162.

de Kleer J., 1986. *Problem solving with the TMS*. Artificial Intelligence, vol. 28, pp. 127–162.

de Kleer J., Williams B., 1987. *Diagnosing multiple faults*. Artificial Intelligence, vol. 32, pp. 97-130, 1987.

de Kleer J., 1989. *Diagnosis with behavioral modes*. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp. 104–109.

de Kleer J., Mackworth A., Reiter R., 1992. *Characterizing diagnoses and systems*. Artificial Intelligence, vol. 56. pp. 197–222.

Fikes R., Hard P., Nilsson N., 1981. *Learning and executing generalized robot plans*. Readings in Artificial Intelligence, pp. 231-249, Palo Alto, Columbia.

Fikes R., Nilsson N., 1971. *STRIPS: A new approach to the aplication of theorem proving to problem solving*. Artificial Intelligence, vol. 2(3-4): 189-208, 1971.

Frank P. M., 1996. *Analytical and qualitative model–based fault diagnosis – a survey and some new results*. European Journal of Control, vol. 2, pp. 6–28.

French S., 1988. *Decision theory. An introduccion to the mathematics of rationality*. Ellis Horwood Limited, 1988.

Friedrich G., 1993. *Theory diagnoses: a concise characterization of faulty systems*. Proc. 13-th International Joint Conference on Artificial Intelligence (IJCAI'93), Chambéry, pp. 1466-1471.

Fuster-Parra P., Ligęza A., 1995. *Qualitative probabilities for causal diagnostic reasoning*. In: *Research and Development in Expert Systems XII* (Bramer, M.A., Nealon, J.L., Milne, R., Eds.). SGES Publications, Information Press Ltd, Oxford, England, pp. 327–340, 1995.

Fuster-Parra P. 1996. *A model for causal diagnostic reasoning. Extended inference modes and efficiency problems.*. PhD Thesis.

Genesereth M. R., 1984. *The use of design descriptions in automated diagnosis*. Artificial Intelligence, vol. 24, pp. 411–436.

Genesereth M. R., Nilsson N.J., 1987. *Logical Foundations of Artificial Intelligence*. M. Kaufmann Publ. Inc., Los Altos, California.

Genesereth M. R., 1993. *From dart to designworld: a chronicle of research on automated engineering in the stanford logic group*. Artificial Intelligence, vol. 59, pp. 159–165.

Gertler J., 1998. *Fault Detection and Diagnosis in Engineering Systems*. New York, Marcel Dekker, Inc.

Giunchiglia F., Walsh T., 1989. *Abstract theorem proving*. Proc. 11th Intl. Joint Conf. on Artificial Intelligence, IJCAI-89, pp. 372–377, Detroit, MI, Morgan Kaufmann.

Giunchiglia F., Villafiorita A., F., Walsh T., 1997. *Theories of abstraction*. Technical Report.

Isermann R., Ballé, P., 1996. *Terminology in the field of supervision, fault detection and diagnosis*. IFAC, SAFEPROCESS.

Isermann R., 1997. *Supervision, fault detection and fault–diagnosis methods - an introduction*. Control Engineering Practice 5(5), pp. 639–652.

Iwasaki Y., Simon H. A., 1986. *Causality in device behavior*. Artificial Intelligence, vol. 29, pp. 3–32.

Karnaugh M., 1953. *The map method for synthesis of combinational logic circuits*. Trans. of AIEE, vol. 72, nr 1, pp. 593–598.

Kleene S. C., 1952. *Introduction to metamathematics*. North Holland, 1952.

Knoblock C. A., Tenenberg J., Yang Q., 1991. *Characterizing abstraction hierarchies for planning*. Proc. Ninth National Conference on Artificial Intelligence, pp 692–697, AAAI, AAAI Press.

Knoblock C. A., 1991. *Automatically generating abstractions for problem solving*. PhD thesis, School of Computer Science.

Knoblock C. A., 1991. *Search reduction in hierarchical problem solving*. Proc. of the 9th National Conference on Artificial Intelligence. AAAI Press.

Kolodner J., 1983. *Reconstructive memory, a computer model*. Cognitive Science, vol. 7, pp. 281-328.

Korbicz J., 1998. *Metody rozpoznawania obrazów w diagnostyce proces'ow przemysłowych*. Mat. III KKN–T Diagnostyka Procesów Przemysłowych, DPP'98. Jurata k. Gdańska, 7–10 września, pp. 93–100.

Korbicz J., *et al.* (Eds.), 2004. *Fault Diagnosis. Models, Artificial Intelligence, Applications*. Springer-Verla, Berlin, Heidelberg, New York.

Kościelny J.M., Szczepaniak P., 1997. *Terminologia oraz klasyfikacja metod detekcji i diagnostyki procesów przemysłowych*. In: Druga Krajowa Konferencja Naukowo–Techniczna Diagnostyka Procesów przemysłowych, pp. 57–68, Łagów, Poland.

Kościelny J.M., S edziak D., Zakroczymski K., 1999. *Fuzzy logic fault isolation in large scale systems*. Int. J. Appl. Math. Comp. Sci., Vol. 9, No. 3, pp. 637–652.

Kościelny J.M., Syfert M., 2000. *Current diagnostics of power boiler system with use of fuzzy logic*. In: Safeprocess'2000, 4th IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes (A. M. Edelmayer, Ed.), vol. 2, pp. 681–686, Budapest.

Kościelny J.M., Bartyś M.Z., 2000. *Application of information system theory for actuators diagnosis*. In: Safeprocess'2000, 4th IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes (A. M. Edelmayer, Ed.), vol. 2, pp. 949–954, Budapest.

Kościelny J.M., 2001. *Diagnostyka zautomatyzowanych procesów przemsłowych*. Warszawa, Akademicka Oficyna Wydawnicza Exit.

Kościelny J.M., 2003. *Modele w diagnostyce procesów*. In: *Diagnostyka procesów* (Korbicz, J., Kościelny, J. M., Kowalczuk, Z., Cholewa, W., Eds.). PWN, pp. 29–56.

Kościelny J.M., 2004. *Diagnostics of industrial processes in decentralised structures*. A Chapter in Korbicz *et al.* (2004), pp. 763–779.

Liebowitz J., 1998. *Applied Expert Systems*. CRC Press.

Ligęza A., 1996. *Completness verification of rule–based control systems*. Systems Analysis – Modeling, Simulation (SAMS) (Int. Journal), vol. 24, pp 211–220.

Ligęza A., 1996. *A logical support for design of complete rule–based systems*. Systems Science, vol. 22, pp. 39–47.

Ligęza A., 2003. *Wybran metody inżynierii wiedzy w diagnostyce systemów*. In: *Diagnostyka procesów* (Korbicz, J., Kościelny, J. M., Kowalczuk, Z., Cholewa, W., Eds.). PWN, pp. 581–622.

Ligęza A., Fuster-Parra P. 1997. *And/or/not casual graphs - a model for diagnostic reasoning*. Applied Mathematics and Computer Science, Vol. 7. No. 1 pp. 57-95.

Ligęza A., Fuster-Parra P. 1998. *A multi–level knowledge–model for diagnostic reasoning* Information Modelling and Knowledge Bases IX (P.-J. Charrel *et al.*, Eds.). IOS Press, Amsterdam, 1998, pp. 330-344.

Milne R., Travé-Massuyés L., 1997. *Model based aspects of the Tiger gas turbine condition monitoring system*. Proc. 3rd IFAC Symp. Fault Detection Supervision and Safety for Technical Processes, SAFEPROCESS'97, R.J. Patton and J. Chen (Eds.), Hull, pp. 420–425.

Mozetič I., 1989. *The role of abstractions in learning qualitative models*. Proc. 4th Intl. Workshop on Machine Learning, pp. 242–255, Irvine, CA, Morgan Kaufmann.

Mozetič I., 1991. *Hierarchical model-based diagnosis*. International Journal of Man-Machine Studies, Vol. 35. pp. 329–362.

Oleksiak J., Ligęza A., 2001. *RuleGraf - program wspomagający projektowanie i weryfikacje poprawności reguł*. MSK - Metody i Systemy Komputerowe w Badaniach Naukowych i Projektowaniu Inżynierskim, Kraków. (in Polish).

Oleksiak J., 2004. *Hierarchical diagnosis in heterogenic environment*. Seminarium: *Przetwarzanie i analiza sygnałów w systemach wizji i sterowania*, Słok. (in Polish).

Oleksiak J., Ligęza A., 2004. *Hierarchical diagnosis of technical systems on the basis of model and expert knowledge*. Recent Developments in Artificial Intelligence methods (eds. T. Burczyński, W. Cholewa and W. Moczulski), AI-METH Series, Gliwice, Poland, pp. 199-203.

Oleksiak J., Ligęza A., 2005. *Hierarchiczna diagnostyka systemów. Model strukturalny i wnioskowanie diagnostyczne*. Diagnostyka Procesów Przemysłowych, DPP'05. (in Polish), (article will be sent).

Oleksiak J., Ligęza A., 2005. *Structural model and reasoning in hierarchical diagnosis*. Journal Computer Assisted Mechanics and Engineering Sciences (CAMES). (article will be sent)

Pawlak Z., 1983. *Systemy informacyjne. Podstawy teoretyczne*. Warszawa, WNT.

Patil R., Szolovits P., Schwartz W., 1981. *Causal understandinag of patient illness in medical diagnosis*. Proc. Int. Join Conf. Artificial Intelligence, IJCAI'81, Vancouver, BC, pp. 893–899.

Patton R.J., Chen J., 1991. *A review of parity space approaches to fault diagnosis*. In: Safeprocess'91, IFAC/IMACS Symposiumon Fault Detection Supervision and Safety for Technical Processes (R. Isermann, Ed.), Baden–Baden.

Pearl J., 1991. *Probabilistic reasoning in intelligent systems: Networks of Plausible Inference*. Morgan Kaufmann Publ. Inc., second printing edition, 1991.

Peirce C. S., 1958. *Collected papers of Charles Sanders Peirce*. Harvard University Press, Vol. 2.

Plaisted D. A., 1980. *Abstraction mappings in mechanical theorem proving*. 5th Conference on Automated Deduction, pages 264–280.

Plaisted D. A., 1981. *Theorem proving with abstraction*. Artificial Intelligence, vol. 16, pp. 47–108.

Ranon R., 2002. *Theories and Techniques of Structural Abstraction for Hierarchical Model–Based Diagnosis*. Ph.D. Thesis.

Reiter R., 1987. *A theory of diagnosis from first principles*. Artificial Intelligence, Vol. 32. pp. 57-95.

Sacerdoti D. E., 1974. *Planning in a hierarchy of abstraction spaces*. Artificial Intelligence, Vol. 5. pp. 115–135.

Sorsa T., Koivo H.N., 1993. *Application of artificial neural networks in process fault diagnosis*. Automatica, vol. 29, no. 4, pp. 843–849.

Strauss P., 1988. *Extensions to ATMS–based diagnosis*. In: J.S. Gero (Eds.), Artificial Intelligence in Engineering, Diagnosis and Learning, Southamptom, pp. 3–28.

Tzafestas S.G., Ed., 1989. *Knowledge–Based System Diagnosis, Supervision and Control*. Plenum Press., New York, London.

Venkatasubramanian V., Rengaswamy R., Kavuri S.N., Yin K., 2003. *A review of process fault detection and diagnosis*. Computers and Chemical Engineering, vol. 27, pp. 293–346.

Zadeh L. A., 1965. *Fuzzy sets*. Inf. Control, Vol. 8. pp. 338–353.

Appendices

Appendix A

Reiter's theory. Calculation of diagnoses

A.0.1 Computing hitting sets

This approach to computing hitting sets is based upon theorem 4.1 and its result is a collection of minimal hitting sets for the collection of conflict sets (Reiter, 1987).

Definition A.1 *Suppose F is a collection of sets. An edge-labeled and node-labeled tree T is an HS-tree for F iff it is the smallest tree with the following properties:*

1. *Its root is labelled by „$\surd$" if F is empty. Otherwise, its root is labeled by a set of F.*
2. *If n is a node of T, define H(n) to be the set of edge labels on the path in T from the root node to n. If n is labeled by $\surd$, it has no successor nodes in T. If n is labeled by a set Σ of F, then for each $\sigma \in \Sigma$, n has a successor node n_σ joined to n by an edge labeled by σ. The label for n_σ is a set $S \in F$ such that $S \cap H(n_\sigma) = \{\}$ if such a set S exists. Otherwise, n_σ is labeled by $\surd$.*

The following results are obvious for any HS–tree for a collection F of sets:

1. If n is a node of the tree labeled by $\surd$, then $H(n)$ is a hitting set for F.
2. Each minimal hitting set for F is $H(n)$ for some node n of the tree labeled by $\surd$.

Example A.1 *Figure A.1 is an HS-tree for F={{2,4,5}, {1,2,3}, {1,3,5}, {2,4,6}, {2,4}, {2,3,5}, {1,6}}.*

Notice, the sets of the form $H(n)$ for nodes labeled $\surd$ do not include all hitting sets for F. The important point for this purpose is that they include all minimal hitting sets for F. The objective is to determine various tree pruning techniques to allow us to generate a subtree of an HS–tree as small as possible, while preserving the property that the subtree will give us all minimal hitting sets for F. In addition, it is suitable to minimize the number of accesses to F required to generate this subtree.

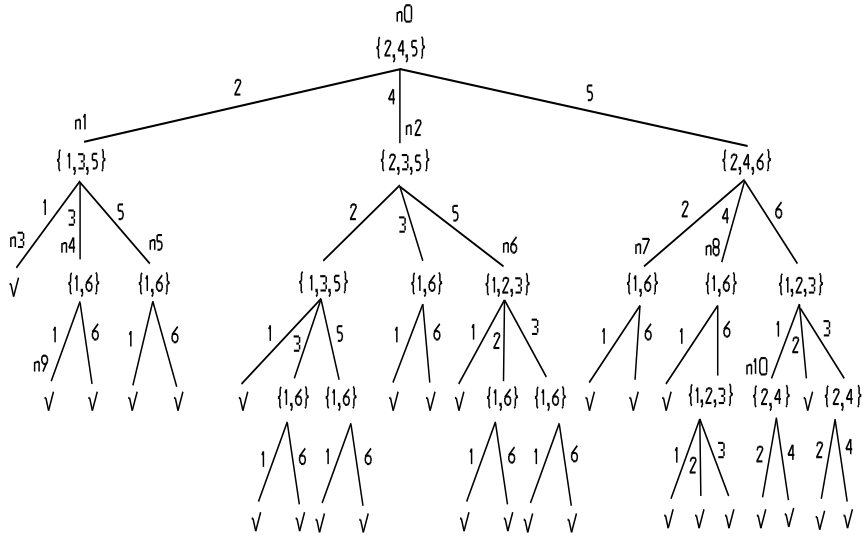

Figure A.1: An HS-tree for F={{2,4,5}, {1,2,3}, {1,3,5}, {2,4,6}, {2,4}, {2,3,5}, {1,6}}

Three lemmas for pruning HS-trees are presented in Reiter (1987). Each of them *should* preserve the property that the resulting pruned HS-tree will include all minimal hitting sets for F.

1. Notice that $H(n_6) = H(n_8)$ (Figure A.1). Moreover, one could have reused the label of n_6 for n_8. This means that the subtrees rooted at n_6 and n_8 respectively could be identically generated had we chosen the reused label for n_8. Thus n_8's subtree is redundant, and we can close node n_8. Similarly, $H(n_7) = H(n_5)$ so we can close node n_7.

2. In Figure A.1, $H(n_3) = \{1, 2\}$ is a hitting set for F. Therefore, any other node n of the tree for which $H(n_3) \subseteq H(n)$ cannot possibly define a smaller hitting set than $H(n_3)$. Since we are only interested in minimal hitting sets, such a node n can be closed. In Figure A.1, node n_9 is an example of such a node which can be closed. The computational advantage of closing node n_9 is that we need not access F to determine that the label of n_9 is $\surd$.

3. The following is a simple result about minimal hitting sets: If F is a collection of sets, and if $S \in F$ and $S' \in F$ with S a proper subset of S', then $F - \{S'\}$ has the same minimal hitting sets as F.

F will be implicitly defined as the set of all conflict sets for $(SD, COMPONENTS, OBS)$. We summarize the method for generating a pruned HS–tree for F as follows:

1. Generate the HS-tree breadth-first, generating nodes at any fixed level in the tree in left-to-right order.

2. Reusing node labels: If node n is labeled by the set $S \in F$, and if n' is a node that $H(n') \cap S = \{\}$, label n' by S. (We indicate that the label of n' is a reused label by underlining it in the tree). Such a node n' requires no access to F.

3. Tree pruning:

   (a) If node n is labeled by $\surd$ and node n' is such that $H(n) \subseteq H(n')$, close n', i.e. do not compute a label for n'; do not generate any successors of n'.

   (b) If node n has been generated and node n' is such that $H(n') = H(n)$, then close n'. A closed node in the tree is indicated by marking it with "$\times$".

   (c) If nodes n and n' have been respectively labeled by sets S and S' of F, and if S' is a proper subset of S, then for each $\alpha \in S - S'$ mark as redundant the edge form node n labeled by α. A redundant edge, together with the subtree beneath it, may be removed from the HS-tree while preserving the property that the resulting pruned HS-tree will yield all minimal hitting sets for F. A redundant edge in a pruned HS-tree is indicated by cutting it with ")(".

It is summarized by the presented below theorem:

Theorem A.1 *Let F be a collection of sets, and T a pruned HS-tree for F, as previously described. Then {H(n) | n is a node of T labeled by $\surd$ } is the collection of minimal hitting sets for F.*

Example A.2 *Figure A.2 is a pruned HS-tree for F={{2,4,5}, {1,2,3}, {1,3,5}, {2,4,6}, {2,4}, {2,3,5}, {1,6}}. The minimal hitting sets are:* $\{1, 2\}$, $\{2, 3, 6\}$, $\{2, 5, 6\}$, $\{4, 1, 3\}$, $\{4, 1, 5\}$, $\{4, 3, 6\}$.

A.0.2 Computing diagnoses

A conceptually simple approach to computing diagnoses can be based upon theorems 4.1 and A.1 as follows: First compute the collection F of all conflict sets for (SD, $COMPONENTS$, OBS), then use the method of pruned HS–trees to compute the minimal hitting sets for F. These minimal hitting sets will be the diagnoses. The problem, then, is to systematically compute all conflict sets for (SD, $COMPONENTS$, OBS). Recall that $\{c_1, \ldots, c_k\} \subseteq COMPONENTS$ is a conflict set iff $SD \cup OBS \cup \{\neg AB(c_1), \ldots, \neg AB(c_k)\}$ is inconsistent. So, using a sound and complete theorem prover, compute all refutations of $SD \cup OBS \cup \{\neg AB(c) | c \in COMPONENTS\}$ and for each such refutation, record the AB instances entering into the refutation. If $\{\neg AB(c_1), \ldots, \neg AB(c_k)\}$ is the set of AB instances used in such a refutation, then $\{c_1, \ldots, c_k\}$ is a conflict set. For example, figure A.3 gives a schematic outline of a resolution style refutation tree for $SD \cup OBS \cup \{\neg AB(c) | c \in COMPONENTS\}$ in which the AB instances entering into the refutation are explicitly indicated. This refutation yields the conflict set $\{c_1, c_5, c_7\}$. Therefore, one approach to computing all conflict sets for (SD, $COMPONENTS$, OBS) is to invoke a sound and complete theorem prover which computes all refutations of $SD \cup OBS \cup \{\neg AB(c) | c \in COMPONENTS\}$, and which, for each such refutation,

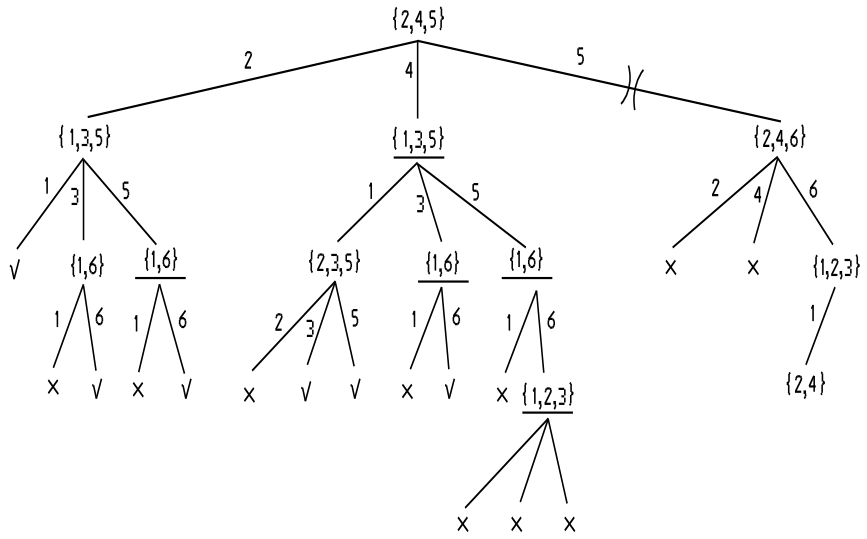

Figure A.2: A pruned HS-tree for F={{2,4,5}, {1,2,3}, {1,3,5}, {2,4,6}, {2,4}, {2,3,5}, {1,6}}

records the AB instances entering into the refutation in order to determine the corresponding conflict set. Unfortunately, there is a serious problem with this approach: the conflict sets do not stand in a 1–1 relationship with the refutations of $SD \cup OBS \cup \{\neg AB(c) | c \in COMPONENTS\}$. There will be refutations which are redundant.

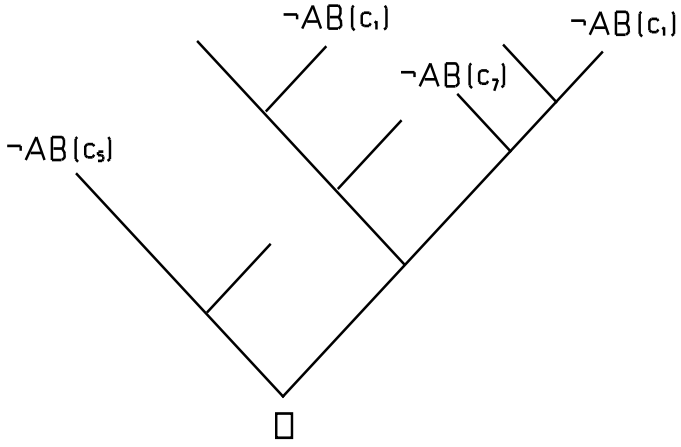

Figure A.3: Resolution style refutation tree for $SD \cup OBS \cup \{\neg AB(c) \mid c \in COMPONENTS\}$ (figure for: $\neg AB(c_1) \wedge \neg AB(c_5) \wedge \neg AB(c_7)$)

An „algorithm" for computing all diagnoses for (SD, $COMPONENTS$, OBS) is developed. This approach is based upon theorem 4.1 and therefore requires all minimal

hitting sets for the collection F of conflict sets for (SD, $COMPONENTS$, OBS). The minimal hitting set calculation will involve generating a pruned HS–tree for F, as per theorem A.1, but with one significant difference: F will not be given explicitly. Instead, suitable elements of F will be computed, as required, while the HS–tree is being generated. Recall that in generating a pruned HS–tree for a collection, F, of sets, a node n of the tree can be assigned a label in one of two ways:

1. By reusing a label S previously determined for some other node n' whenever $H(n) \cap S = \{\}$; in this case, no access to F is required since the label for n is obtained from that part of the pruned HS–tree generated thus far.

2. By searching F for a set S such that $H(n) \cap S = \{\}$. If such a set S can be found in F, n is labeled by S, otherwise by $\surd$. In this case the set F must be accessed; n's label can not be determined without F. Now it should be clear that the set F need not be given explicitly. The only time that F is needed is in case 2) above. Therefore, to generate a pruned HS–tree for F, we only require a function which, when given $H(n)$, returns a set S such that $H(n) \cap S = \{\}$ if such a set S exists in F, and $\surd$ otherwise. We now exhibit such a function when F is the collection of conflict sets for (SD, $COMPONENTS$, OBS). Let $TP(SD, COMPONENTS, OBS)$ be a function with the property that whenever $(SD, COMPONENTS)$ is a system and OBS an observation for that system, $TP(SD, COMPONENTS, OBS)$ returns a conflict set for $(SD, COMPONENTS, OBS)$ if one exists, i.e. if $SD \cup OBS \cup \{\neg AB(c) | c \in COMPONENTS\}$ is inconsistent, and returns $\surd$ otherwise. It is easy to see that any such function TP has the following property: If $C \subseteq COMPONENTS$, then $TP(SD, COMPONENTS - C, OBS)$ returns a conflict set S for $(SD, COMPONENTS, OBS)$ such that $C \cap S = \{\}$ if such a set exists, and $\surd$ otherwise. It follows that we can generate a pruned HS–tree for F, the collection of conflict sets for $(SD, COMPONENTS, OBS)$ as described in a former section except that whenever a node n of this tree needs an access to F to compute its label, we label n by $TP(SD, COMPONENTS - H(n), OBS)$. From this pruned HS–tree T we can extract the set of all minimal hitting sets for F, namely $\{H(n) | n$ is a node of T labeled by $\surd\}$. By theorem A.1, this is the set of diagnoses for $(SD, COMPONENTS, OBS)$.

The final algorithm can be written as procedure $DIAGNOSE$(SD, $COMPONENTS$, OBS) where (SD, $COMPONENTS$) is a system and OBS is an observation of the system. TP is any function with the property that TP(SD, $COMPONENTS$, OBS) returns a conflict set for (SD, $COMPONENTS$, OBS) if one exists, i.e. if $SD \cup OBS \cup \{\neg AB(c) | c \in COMPONENTS\}$ is inconsistent, and returns $\surd$ otherwise. $DIAGNOSE$(SD, $COMPONENTS$, OBS) returns the set of all diagnoses for (SD, $COMPONENTS$, OBS).

- Step 1. Generate a pruned HS–tree T for the collection F of conflict sets for $(SD, COMPONENTS, OBS)$ as described in a former section except that whenever, in the process of generating T a node n of T needs an access to F to compute its label, label that node with $TP(SD, COMPONENTS - H(n), OBS)$.

- Step 2. Return $\{ H(n) | n$ is a node of T labeled by $\surd\}$.

Appendix B

Approaches to hierarchy

This appendix presents a few good known approaches to hierarchy. Their original terminology is kept. A more thorough survey of abstraction methodologies and applications is presented in a few documents e.g. (Giunchiglia *et al.*, 1997), (Ranon, 2002).

B.0.3 Theory of abstraction

In (Giunchiglia and Walsh, 1989), authors present theoretical bases of proving in abstract–detailed systems. Abstraction is there informally defined as a process of mapping the original representation of a problem, called a *ground* representation, onto a new representation, called a *abstract* representation. More precisely, there is defined a formal system (Kleene, 1952) as a formal description of a theory or a problem, and then abstraction as a pair of such systems.

Definition B.1 (Giunchiglia & Walsh) *A formal system Σ is a triple $\langle \Lambda, \Delta, \Omega \rangle$, where Λ is the Language, Ω is the set of axioms and Δ is the Deductive Machinery of Σ.*

Definition B.2 (Giunchiglia & Walsh) *An abstraction, written $f : \Sigma_1 \Rightarrow \Sigma_2$ is a pair of formal systems $\langle \Sigma_1, \Sigma_2 \rangle$ with languages Λ_1 and Λ_2 respectively, and an effective total function $f_\Lambda : \Lambda_1 \rightarrow \Lambda_2$.*

Σ_1 is there called a *ground* space and Σ_2 an *abstract* space. f_Λ is called a mapping function. $f_\Lambda : \Sigma_1 \Rightarrow \Sigma_2$ means that $\langle \Sigma_1, \Sigma_2 \rangle$ and f_Λ constitute an abstraction. The authors distinguish there a few types of abstraction. Two of them, TI and NTI abstractions, are useful in diagnosis and they are described below.

An abstraction $f_\Lambda : \Sigma_1 \Rightarrow \Sigma_2$ is said to be a TI abstraction (Theorem Increasing abstraction) iff, for any well formed formula α, if $\alpha \in TH(\Sigma_1)$ then $f_\Lambda(\alpha) \in TH(\Sigma_2)$. $TH(\Sigma)$ is the set of theorems, wffs, of a formal system Σ. This property is illustrated in figure B.1. A TI abstract space $TH(\Sigma_2)$ is broader than TI ground space $TH(\Sigma_2)$.

Some of solutions from the abstract space can be not valid for the ground space, but generation of solutions on the abstract space and verification on the ground space is less computationally costly than generation of solutions directly for the ground space. The TI abstraction allows to relax a system description on the abstract level what can be understand as relaxation of variables, potential values of variables, constraints and so on.

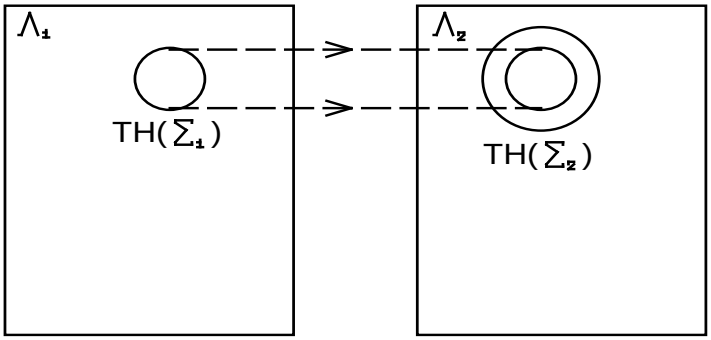

Figure B.1: TI–abstraction (Giunchiglia and Walsh, 1989)

The second useful abstraction is NTI (Non Theorem Increasing abstraction). An abstraction $f : \Sigma_1 \Rightarrow \Sigma_2$ is said to be a NTI abstraction iff, for any wff α, if $\alpha \in NTH(\Sigma_1)$ then $f_\Lambda(\alpha) \in NTH(\Sigma_2)$. NTH($\Sigma$) is a set of wffs that if added as an assumption to Σ make the resulting system inconsistent. This kind of abstraction is useful for methodologies based on refutation. The idea of NTI abstraction is illustrated in figure B.2.

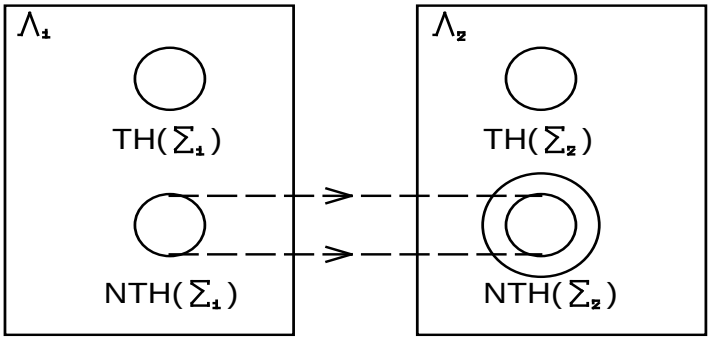

Figure B.2: NTI–abstraction (Giunchiglia and Walsh, 1989)

B.0.4 ABSTRIPS

An another hierarchical approach is ABSTRIPS framework (Sacerdoti, 1974). The framework is based on STRIPS („Stanford Research Institute Problem Solver") planning system (Fikes and Nilsson, 1971) and (Fikes *et al.*, 1981).

Definition B.3 (Fikes) *A STRIPS system S is a triple (L, s_0, O) where L is a language, s_0 is the initial state, and O is the set of operators.*

Definition B.4 (Fikes) *An operator Op of a STRIPS system (L, s_0, O) is a triple $(P_\alpha, D_\alpha, A_\alpha) \subset O$, where $P_\alpha \subset L$ is the precondition list, $D_\alpha \subset L$ is the delete list, and A_α is the add list.*

The STRIPS operators map states to states. We can say generally that when operator preconditions are satisfied then the operator can be used and sentences from the *delete list* are deleted from the system state, sentences from *add list* are added.

STRIPS approach is designed for planning, in other words for finding a finite sequence of operators which transform an initial state to a goal state. ABSTRIPS framework creates abstraction by selective dropping of preconditions of operators for STRIPS system. ABSTRIPS abstraction is created only by weakening operators. Dropping of preconditions is done according to their criticality what corresponds to difficulty of satisfying them.

Knoblock proposes additional function *crit* which assigns values to each precondition of each operator (Knoblock *et al.*, 1991). The value is proportional to the length of a plan necessary to achieve a precondition, in other words to number of operators necessary to achieve the precondition. Abstraction of a system is on a *k-level* if all evaluated below *k* preconditions are dropped.

Modeling of larger systems, technological or others, is not easy when only operators can be weak and it is the biggest disadvantage of this approach. There are a few other systems which are close to ABSTRIPS e.g. ALPINE (Knoblock, 1991), HIGHPOINT (Bacchus and Yang, 1994) or (Bundy and Giunchiglia *et al.*, 1996).

B.0.5 Clauses abstraction

Plaisted in papers (Plaisted, 1980) and (Plaisted, 1981) presents abstraction based on subsumption between first–order clauses. The abstraction has to fulfill a condition that if C_3 is a resolvent of two clauses C_1 and C_2 and they have abstractions $f(C_3)$, $f(C_1)$, $f(C_2)$ then some resolvent of $f(C_1)$, $f(C_2)$ has to subsume $f(C_3)$. If the condition is fulfill, what is not easy, then resolution theorem proving is possible for such abstract clause system.