Constraint-Driven Fault Diagnosis

Rafael M. Gasca, Ángel Jesús Varela-Vaca and Rafael Ceballos

14.1 Introduction

Constraint-Driven Fault Diagnosis (CDD) is based on the concept of *constraint suspension* [6], which was proposed as an approach to fault detection and diagnosis. In this chapter, its capabilities are demonstrated by describing how it might be applied to hardware systems. With this idea, a model-based fault diagnosis problem may be considered as a Constraint Satisfaction Problem (CSP) in order to detect any unexpected behavior and Constraint Satisfaction Optimization Problem (COP) in order to identify the reason for any unexpected behavior because the parsimony principle is taken into account.

In order to automate the CDD, in an efficient way, different techniques for solving these CSP/COPs could be considered. The first to be considered is related to the mathematical properties of the semiring CSP [26], where efficient diagnosis solutions are obtained decomposing the problem into trees applying dynamic programming. The second one is related to the solving of overconstrained CSPs and identification of conflicts [1, 20, 24]. The third approach [14, 15] uses a known symbolic technique called as Gröbner basis, that allows obtaining automatically, in certain cases, the Analytic Redundancy Restrictions which are used in Fault Detection and Identification (FDI) paradigm. The last efficient technique is related to specific clustering techniques that allow decomposing the original model-based diagnosis in several simpler problems and the computational complexity is reduced in a significant way.

R. M. Gasca · Á. J. Varela-Vaca · R. Ceballos (✉)
Department of Computer Languages and Systems,
Universidad de Sevilla, Seville, Spain
e-mail: ceball@us.es

R. M. Gasca
e-mail: gasca@us.es

Á. J. Varela-Vaca
e-mail: ajvarela@us.es

In this chapter, the CDD is applied to typical hardware diagnostic problems such as multiplier–adder circuits and heat exchangers.

14.2 Constraint Programming in a Nutshell

Constraint Programming is based on the automatic resolution of CSPs, which are problems where an assignment of values to variables must be found in order to satisfy a finite number of constraints.

The model-based approach for automating the fault detection problems may be expressed as a Boolean Satisfiability Problem (SAT) or CSP instances. Therefore, the solving process consists of trying to run an SAT or CSP solver, respectively, and to determine whether they have any solution. In the case that the problems have no solution, the automation of the fault diagnosis problems consist of solving a MAX-SAT or MAX-CSP (COP) in order to implement constraint suspension.

A good guideline or rule to choose SAT or CSP representation can be related to the natural expression of the hardware models for using in model-based diagnosis approach. Another rule could be related to the computational efficiency, because either SAT solvers perform better than CSP solvers or where CSP solvers perform better than SAT solvers.

14.2.1 CSP and COP

On top of constraint technology lies the concept of the CSP [10]. In general, a CSP is composed of a set of variables, a domain for each variable, and a set of constraints. Each constraint is defined over some subset of the original set of variables and limits the combinations of values that the variables in this subset can take. The goal is to find one assignment to the variables such that the assignment satisfies all the constraints. In some kind of problems, the goal is to find all such assignments [23]. The constraints are then the relationships between the various choices for the variables.

Definition 14.1 A Constraint Satisfaction Problem (CSP) represents a reasoning framework consisting of variables, domains and constraints $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$. The set of variables $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ could have the following domains $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ and a constraint c_k $(v_{k_1}, \ldots, v_{k_n})$ is the set of assignment of the subset of variables in the corresponding domains that belongs to Cartesian Product $d_{k_1} \times \cdots \times d_{k_n}$. The solutions of a CSP is the set of assignment of the variables $\mathcal{V}$ in their corresponding domains that satisfy all constraints $\mathcal{C}$.

A typical example of CSP is the map-coloring problem, where given a geographical map split into regions, a color has to be assigned to all the regions dealing with adjacent regions must have different colors. An example map with eight regions is depicted in Fig. 14.1. The formalization of this CSP could be as follows:

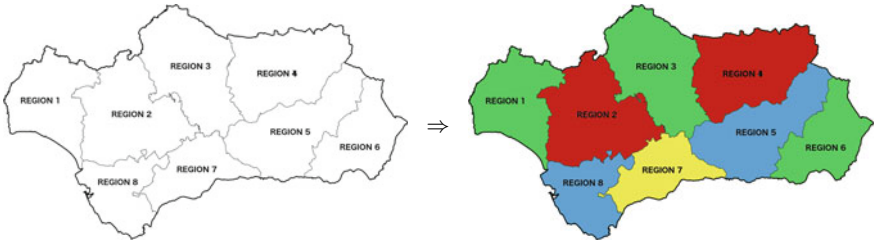


Fig. 14.1 Map-coloring problem

$$\mathcal{V} = \{R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8\}. \tag{14.1}$$

$$\mathcal{D} = \{red, green, blue, yellow\}. \tag{14.2}$$

$$\mathcal{C} = \{R_1 \neq R_2, R_1 \neq R_8, R_2 \neq R_3, R_2 \neq R_8, R_2 \neq R_7, R_3 \neq R_4, R_3 \neq R_5, R_7 \neq R_5, R_5 \neq R_4, R_5 \neq R_6\}. \tag{14.3}$$

The problem is composed of eight variables that represent each region, the domains are the colors, and the constraints establish the inequality conditions that the color of one region is different from the adjacent regions.

One application of the CSP might be the determination of the color assignments to regions, for instance, as given in Fig. 14.1. Another application might be given a solution to determine whether the solution is satisfiable or to determine the fault diagnosis pointing out the constraint and variables that make the CSP unsatisfied. This type of problem is modeled as MAX-CSP, thus a Constraint Optimization Problem (COP).

Definition 14.2 A Constraint Optimization Problem (COP) is a CSP in which the solutions optimize (minimize or maximize) an objective function, f over a subset of variables $\mathcal{V}$ of the CSP.

Following the map-coloring example, if a color assignment is given as observational model such as the right map as shown in Fig. 14.2. By applying the previous CSP, we can check the unsatisfiability since the regions R_1 and R_8 have the same color *green*. In this case, reified constraints can be applied by means of a COP (i.e., MAX-CSP) in order to automatically determine the faults. The representation of the problem might be as follows:

$$\mathcal{V} = \{R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, RR_1, RR_2, RR_3, RR_4, RR_5, RR_6, RR_7, RR_8, RR_9, RR_{10}\}. \tag{14.4}$$

$$\mathcal{D}_{R_x} = \{red, green, blue, yellow\}. \tag{14.5}$$

$$\mathcal{D}_{RR_x} = \{0, 1\}. \tag{14.6}$$

$$\mathcal{C} = \{R_1 = green, R_2 = red, R_3 = green, R_4 = red, R_5 = blue,$$

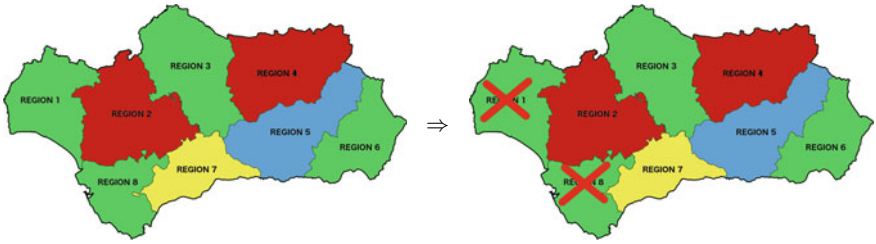


Fig. 14.2 Fault diagnosis of an assignment

$$R_6 = green, R_7 = yellow, R_8 = green. \quad (14.7)$$

$$RR_1 = 1, RR_2 = 1, RR_3 = 1, RR_4 = 1, RR_5 = 1, RR_6 = 1,$$
$$RR_7 = 1, RR_8 = 1, RR_9 = 1, RR_{10} = 1, \quad (14.8)$$

$$RR_1 == R_1 \neq R_2, RR_2 == R_1 \neq R_8, RR_3 == R_2 \neq R_3,$$
$$RR_4 == R_2 \neq R_8, RR_5 == R_2 \neq R_7, RR_6 == R_3 \neq R_4,$$
$$RR_7 == R_3 \neq R_5, RR_8 == R_7 \neq R_5, RR_9 == R_5 \neq R_4,$$
$$RR_{10} == R_5 \neq R_6\}. \quad (14.9)$$

$$\mathcal{F} = maximize(RR_1 + RR_2 + RR_3 + RR_4 + RR_5 + RR_6 +$$
$$RR_7 + RR_8 + RR_9 + RR_{10}). \quad (14.10)$$

The COP tries to maximize the sum of the reified constraints to be satisfied. The reified constraints are represented as "Boolean" variables RR_x with zero or one as domain. The RR_x variables have been forced to 1 value (cf. $RR_x = 1$). These values force the accomplishment of the constraints such as $RR_1 == R_1 \neq R_8$. However, this constraint cannot be satisfiable since the values of R_1 and R_8 are the same, *green*. This contradiction makes the COP unsatisfiable due to the assignment of *green* color for the adjacent regions *Region 1* and *Region 8* as shown in right map in Fig. 14.2.

The techniques used in solving the CSPs depend on the kind of constraints being considered. Constraints are often used on a finite domain, to the point that CSP is typically identified with problems based on constraints on a finite domain. Such problems are usually solved via techniques that combine propagation and search, in a particular way of backtracking or local search. Constraint propagation is a method used on such problems, but the majority of them are incomplete. In general, they may solve the problem or check if it is unsatisfiable, but not always. For this reason, these methods are also used in conjunction with a search algorithm to make possible the solving of these problems. Other considered kinds of constraints are on real or rational numbers, solving problems on these constraints is done via specific constraint propagation and search algorithms.

One of the main difficulties in CSP resolution is the appearance of local inconsistencies. Local inconsistencies are values of the variables that cannot take part in the solution because they do not satisfy any consistency property. Therefore if any con-

sistency property is forced, we can remove all the values which are inconsistent with regard to the property. But it can be possible that some values which are consistent with regard to a property are inconsistent with regard to another property at the same time. Global consistency implies that all values which cannot take part in a solution can be removed. The constraints of a CSP generate local inconsistencies because they are combined. If the search algorithm does not store these inconsistencies, it will waste time and effort trying to carry out instantiations which have already been tested.

Different orders of consistency can be considered [22] according to the size of subnetworks taken into account: 1-consistency for networks involving one variable, 2-consistency for two variables, and in general k-consistency for networks involving k variables. An algorithm for computing k-consistency for discrete CSP [12] had been proposed whose runtime is exponential in k.

As it has been shown, since CSP can be used to detect constraint inconsistencies, it can be also be used within the model-based diagnosis paradigm. To diagnose a system within the CSP framework one must first represent the model of the system structure and the correct behavior. The structural elements of the system being diagnosed are modeled as CSP variables, and the behavioral relationships among the constituent components are expressed as constraints.

The diagnosis schema employs a CSP engine, whose outputs are the expected diagnoses based on the discrepancies between the predictions of the CSP representation and the observations from running the system. The diagnosed system is found faulty if some constraints cannot be satisfied, in which case the violated constraints are the precise reasons for the cause of the fault [11].

14.2.2 SAT and MAX-SAT

The Boolean Satisfiability Problem (SAT) is the problem of deciding whether there is a variable assignment that satisfies a given propositional expression. The boolean expressions with Boolean variables have no quantifiers. In 3-SAT, the maximum number of literals for the clause is 3 and in 2-SAT, the maximum number of literals for the clause is 2. It is solved efficiently by using path searches in graphs. For certain problems of fault detection, the SAT can be used and the MAX-SAT for fault diagnosis. An application of SAT and MAX-SAT is going to be illustrated in the Model-based Software Debugging chapter.

The solving process of these problems is carried out by means of SAT Solver or either translating MAX-SAT to MAX-CSP that is trivial. Translating CSP to SAT is not trivial, because it is necessary to convert multivalued variables into a set of boolean assignment variables and translate the constraints into clauses over assignment variables. SAT solvers may be complete such as solvers based on the Davis–Putnam–Logemann–Loveland (DPLL) algorithm, and incomplete SAT solvers based on local search and hybrid SAT solvers.

14.3 Automation of Constraint-Driven Fault Diagnosis

In order to efficiently automate the constraint-driven fault diagnosis, a set of concepts is necessary to consider for a system with m components (physical components):

Definition 14.3 *System Model (SM)* is a finite set of linear or nonlinear (polynomial) equality constraints ($\mathcal{P}$), which determine the hardware system behavior. This is done by means of the relations between the system non-observable variables ($\mathcal{V}_{nob}$) and the system observable variables ($\mathcal{V}_{ob}$), which are directly obtained from sensors that are supposed to work correctly for hardware systems. The representation of an SM is a tuple ($\mathcal{P}$, $\mathcal{V}_{ob}$, $\mathcal{V}_{nob}$).

An example used in the bibliography with respect to model-based diagnosis [5–7, 16, 22] is the polybox system previously introduced in Chap. 2 (cf. Logical Case Studies). This system is composed of three multipliers and two adders such as the shown in Fig. 14.3. We will have the following System Model (SM): $\langle \mathcal{P} : \{a * c = x, b * d = y, c * e = z, x + y = f, y + z = g\}, \mathcal{V}_{ob}\{a, b, c, d, e, f, g\}, \mathcal{V}_{nob} : \{x, y, z\}\rangle$.

Definition 14.4 *Context Set (CS)* is a subset of components of the system and its associated constraints. The possible context set will be 2^m.

Definition 14.5 *Context Network (CN)* is a lattice of subsets of the component of the system, ordered by "is subset of", according to the way proposed by ATMS [21]. The number of possible contexts for a system of n components is $2^m - 1$.

Definition 14.6 *Observational Model (OM)* is defined by a tuple of values for the observable variables. In the example model in Fig. 14.3, two observation models could be

$$OM_1 \equiv \{a = 3, b, = 2, c = 2, d = 3, e = 3, f = 10, g = 12\}. \quad (14.11)$$

$$OM_2 \equiv \{a = 3, b = 2, c = 2, d = 3, e = 3, f = 10, g = 10\}. \quad (14.12)$$

Definition 14.7 *Diagnosis Problem* is defined by a tuple composed of a System Model (SM) and an Observational Model. The solution to this problem is a set of possible failed components of the system.

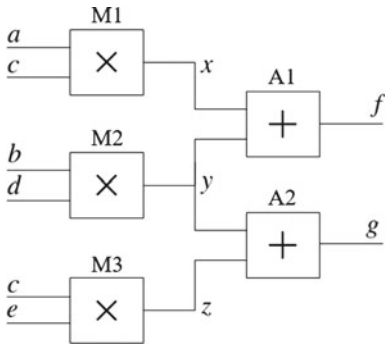


Fig. 14.3 Model of three multipliers and two adders

A set of computational techniques for the constraint-driven fault diagnosis are presented in the following subsections.

14.3.1 Constraint Suspension for Diagnostic Reasoning

In 1984, Davis [6] presents a new technique for diagnostic reasoning called constraint suspension. The reasoning from first principles is proposed using different languages for describing the structure and the component behavior. According to these languages, a constraint network is obtained and then the goal is to derive the components that could be responsible for a concrete fault, given the observations of the system. In the faulty case, the constraint network is inconsistent and the possible consistency is achieved by retracting some constraints (component behavior) of the network when we take into account the observational model. In order to improve the efficiency, this work proposed the concept of *discrepancy detection*.

14.3.2 Semiring CSP for Diagnostic Reasoning

Semiring CSP [2] is a framework for soft constraints. Soft constraints establish preference levels changing the definition of hard constraints. In these soft constraints, the assignments are associated with different interpretations such as weight, cost, preference, etc.

Model-based diagnosis may be solved as a COP over lattices. Sachenbacher and Williams [26] presents a framework as semiring CSPs. The mathematical properties of a semiring CSP allow to obtain efficient solution decomposing model-based problems into trees and later applying dynamic programming. The method allows to perform diagnosis over the general class of lattice preference structures and it is compared with the diagnosis algorithms SAB [9] and TREE* [27] for tree-structured problems that are special cases.

14.3.3 Overconstrained CSPs and Identification of Conflicts

Solving CSP process assigns values to variables satisfying all of the set of constraints, but it is impossible when a fault is present in the diagnosis problems. In the constraint programming area, these problems are called as overconstrained CSP and do not have any solution. We need to consider some ways of relaxing or weakening the specification of detection fault problem (in our case of the description of the hardware system model) until solutions can be found. Therefore, the systematic or local search does not find solutions for these overconstrained CSPs. Nevertheless, the number of

satisfied constraints could be maximized for the detection fault problem. There are in the bibliography four ways of weakening an overconstrained CSP:

- Extending the domain of a variable,
- Extending the definition of a constraint,
- Eliminating any variable,
- Suspending any constraint.

The idea, related to fault diagnosis, is that given an infeasible set of constraints C, the goal is obtain explanations of infeasibility. For this reason, we must find out:

- All Minimal Unsatisfiable Subsets (MUSes) of the problem where all proper subsets satisfiable or,
- All Maximal Satisfiable Subsets (MSSes) of the problem.

The definition and strong relationship between MUSes and MSSes of a determined overconstrained problem has been analyzed in a previous work [25]. An adequate and specific solver could compute all Minimally Unsatisfiable Subsets (MUSes) of a given overconstrained CSP. For the following overconstrained CSP:

$$\mathcal{V} = \{a, b, c, d, e, f, g, x, y, z\}. \tag{14.13}$$

$$\mathcal{D} = \{2, 3, 3, 2, 2, 10, 12, [0, 100], [0, 100], [0, 100]\}. \tag{14.14}$$

$$\begin{aligned} \mathcal{C} = \{c_1 = a \times c = x, \quad & c_2 = b \times d = y, c_3 = c \times e = z, \\ & c_4 = x + y = f, c_5 = y + z = g\}. \end{aligned} \tag{14.15}$$

The set of constraints $\{c_1, c_2, c_4\}$ and $\{c_1, c_3, c_4, c_5\}$ are minimally unsatisfiable subsets for this problem. They will eventually be used for obtaining the solution of the fault diagnosis problem. A CSP is satisfiable iff it contains no MUSes.

Also, we could determine MSSes or CoMSSes, that is the complement of MSSes. Every CoMSS could be obtained as an irreducible hitting set of the collection of all MUSes. In the above example, CoMSSes are $\{c_1\}$, $\{c_4\}$, $\{c_2, c_3\}$, $\{c_2, c_5\}$. Therefore, hitting sets provide a transformation from MUSes to CoMSSes and they provide the possible solutions for the fault diagnosis problem. This transformation removes those constraints making the overconstrained CSP unsatisfiable.

Also, in the literature, Baker et al. [1] propose the Diagnosis of Over-determined CSPs (DOC), which identifies the set of least-important constraints that should be relaxed to the initial overconstrained CSP has a solution. If the solution is unacceptable for a user, the DOC selects next-best sets of least-important constraints until an acceptable solution has been generated. QUICKXPLAIN [20] improves this method by decomposing the complete explanation into subproblems of the same size.

For overconstrained CSP in specific domains, different works have been proposed in the bibliography. In overconstrained instances of the Disjunctive Temporal Problem (DTP), Liffiton et al. [24] describe the Musilitis algorithm for finding MUSes of

overconstrained DTP. Also, for numeric overconstrained CSPs (NCSPs), the paper [13] proposes a set of methods for efficiently deriving all Numeric MUSes (NMUS) using the Neighborhood-based Structural Analysis on overconstrained NCSP. In such work, different bottom-up derivation strategies are described by taking into account the concept of the neighborhood for the different types of NCSPs. Depending on the structural aspects of these problems, the search methods are different.

14.3.4 Symbolic Techniques

In engineering problems, the models are almost always a set of linear and polynomial constraints. In order to automate and improve the fault diagnosis a new model can be derived, which is made up of a single constraint using a symbolic technique such as Gröbner basis. First, it eliminates the non-observable variables and obtains new constraints of the different set of components of the system. We build a context network with these polynomial constraints in the node and propose a methodology that is composed of an incremental algorithm that avoids the recalculation of previous constraints. A standard framework is used to obtain the minimal diagnosis. The obtained results are similar to those shown in the bibliography, but they are obtained by means of a more efficient and automatic way. This approach may be very useful for on-board diagnosis [14, 15].

Before presenting the methodology to carry out the fault diagnosis process for this model, we need to present the concept of Gröbner basis. A Gröbner basis is a set of multivariate polynomials that have desirable algorithmic properties. An algorithm permits to transform the set of polynomials into a Gröbner basis. It generalizes previous techniques such as Gaussian elimination for solving linear systems of equations, the Euclidean algorithm for computing the greatest common divisor of two univariate polynomials, and the Simplex Algorithm for linear programming. The introduction for Gröbner basis is presented in several works [3, 8, 19].

According to theoretical considerations, a set of multivariate polynomials $\mathcal{P} = 0$ does not have solutions if and only if 1 is in its Gröbner basis [3]. Therefore,

- If a polynomial model is overconstrained and have redundant constraints, the calculation of Gröbner basis eliminated them.
- If the polynomial model is overconstrained and inconsistent, one of the constraints obtained is $1 = 0$, that determine the inconsistency of the model.

Table 14.1 Minimal conflict contexts of the model in Fig. 14.3 without observational model

Minimal conflict contexts	Associated constraints	Truth value
M_1, M_2, A_1	$a * c + b * d - f = 0$	Not evaluated
M_2, M_3, A_2	$b * d + c * e - g = 0$	Not evaluated
M_1, M_3, A_1, A_2	$a * c - c * e - f + g = 0$	Not evaluated

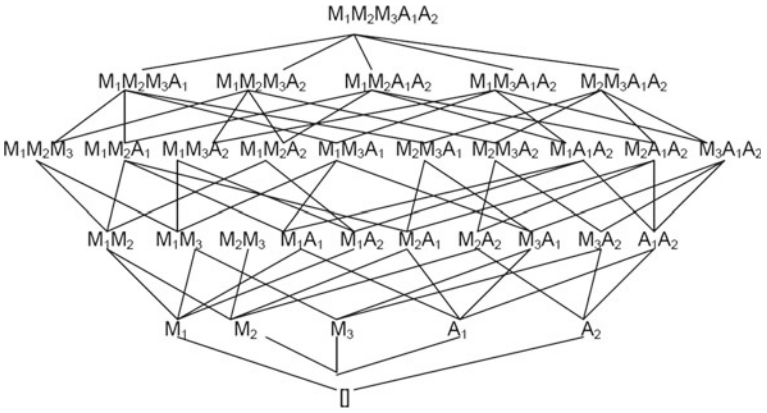


Fig. 14.4 Context network for the model of three multipliers and two adders

- If a set of polynomials is underconstrained, the Gröbner basis obtained is very useful for the possible resolution.

The parameters of the function *GröbnerBasis* are the set of polynomial constraints, set of observable variables, set of non-observable variables. These functions allow the calculation of the Gröbner basis. For example in Fig. 14.4, given the context for the components $\{M_1, M_2, M_3, A_1, A_2\}$, the function GröbnerBasis $(\{a * c - x, b * d - y, c * e - z, x + y - f, y + z - g\}, \{a, b, c, d, e, f, g\}, \{x, y, z\})$ obtains the following result:

$$\{b * d + c * e - g = 0, a * c - c * e - f + g = 0\}. \tag{14.16}$$

The application of the *GröbnerBasis* function to all contexts of the system represented in Fig. 14.3 permits to obtain the new context network represented in Table 14.1 and Fig. 14.5. In this figure, the contexts with only one component are not represented because all the contexts with two components of the Gröbner function are empty. Only the contexts with no-empty Gröbner functions are relevant and useful for fault diagnosis.

According to this new context network, the methodology for solving the fault diagnosis problem for the system given an observation model is the following:

- Searching the Minimal Conflict Contexts: An algorithm of Breadth-First Search (BFS) of the graph represented in Fig. 14.5 obtain for this system:
- Considering the observational model OM_1 the previous table is completed. We obtain the minimal conflict contexts (ARR in FDI) that are evaluated as false, resulting as shown in Table 14.2.
- Obtaining the minimal diagnoses. A standard algorithm to obtain the hitting set is used. For the previous observational models, the results are presented in Table 14.3.

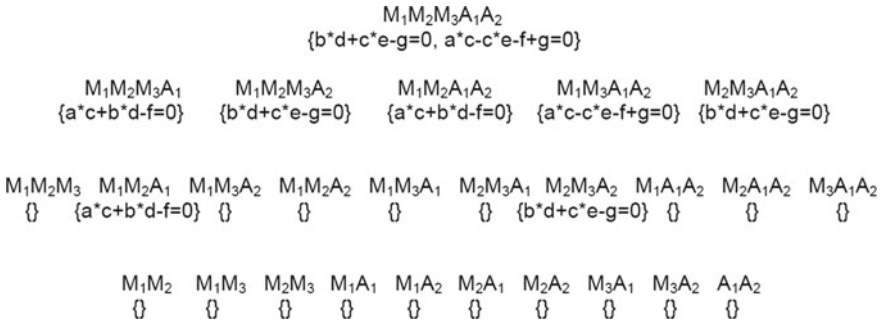


Fig. 14.5 Context network with Gröbner Basis for the model of three multipliers and two adders

Table 14.2 Minimal conflict contexts of the model in Fig. 14.3 with observational model OM_1

Minimal conflict contexts	Associated constraints	Truth value
M_1, M_2, A_1	$a*c+b*d-f=0$	False
M_2, M_3, A_2	$b*d+c*e-g=0$	True
M_1, M_3, A_1, A_2	$a*c-c*e-f+g=0$	False

Table 14.3 Minimal diagnoses of the model in Fig. 14.3

Observational model	Minimal diagnosis
OM_1	{A1}, {M1}, {A2, M2}, {M2, M3}
OM_2	{M2}, {A1, A2}, {A1, M3}, {A2, M1},{M1, M3}

As an exercise, the reader can try to build the Minimal Conflict Context for OM_2 similarly to the Table 14.2. Finally, compare the results with the diagnosis in Table 14.3.

14.3.5 Improvements for Determining the Minimal Conflicts Contexts

For determining the Minimal Conflicts Contexts in an efficient way, in this subsection two improvements are proposed [4]: a structural pretreatment in order to reduce drastically the number of nodes of context network, and a reduction of the number of contexts where Gröbner Bases algorithm is applied. These two improvements can be done in an offline process.

14.3.5.1 Clusters of Components

The first step is the partition of the system into independent subsystems, in such a way that the Minimal Conflicts Contexts of the system can be obtained as the union of all Minimal Conflicts Contexts of all subsystems. These subsystems are smaller than the whole system, and therefore the computational complexity for detecting conflicts is reduced.

Definition 14.8 *Cluster of components*. A set of components $\mathcal{C}$ is a cluster of component only if:

1. For all non-observable inputs and outputs of each component of $\mathcal{C}$, these inputs and outputs are always linked only with components of $\mathcal{C}$.
2. It does not exist another set $\mathcal{C}' \subseteq \mathcal{C}$ which satisfies the first condition.

The first part the definition guarantees that we are able to detect a conflict in a cluster of components without information about other clusters. The second part of the definition guarantees that the size of clusters will be as small as possible. For each cluster, an independent context network is obtained. The set of conflicts for each cluster can be obtained by independent processes (also in a parallel way), and the union of the conflicts of all the clusters will be the complete set of conflicts of the system.

Example There is just one cluster in the polybox example, and the context network contains $2^5 - 1 = 31$ nodes. But, for example, if the visibility of variable y is changed from non-observable to observable, there will be three clusters: $\{M_1, A_1\}$, $\{M_2\}$ and $\{M_3, A_2\}$, and there will be three context networks with a total of $(2^2 - 1) + (2^1 - 1) + (2^2 - 1) = 7$ nodes.

14.3.5.2 Relevant Contexts

In order to obtain an equivalent context network but without non-observable variables, Gröbner Basis algorithm is applied. The goal is reducing the number of nodes (of the context network) to be processed, and therefore reducing the computational complexity. The idea is to select which contexts are important for detecting conflicts. These contexts are named relevant contexts.

Definition 14.9 *Relevant contexts*. RC is a relevant context if applying Gröbner Basis algorithm (GB), one or more obtained constraints cannot be obtained by applying the same algorithm to other sub-contexts $RC' \subseteq RC$, and $\forall\, RC' \subseteq RC$: $GB(RC') \subseteq GB(RC)$.

In order to know if a context is relevant or irrelevant, the first step is removing for each context all the constraints which include only observable variables. In this case, it has no sense to apply Gröbner Basis algorithm. The next step is removing the constraints that contain at least one non-observable variable, which appears only

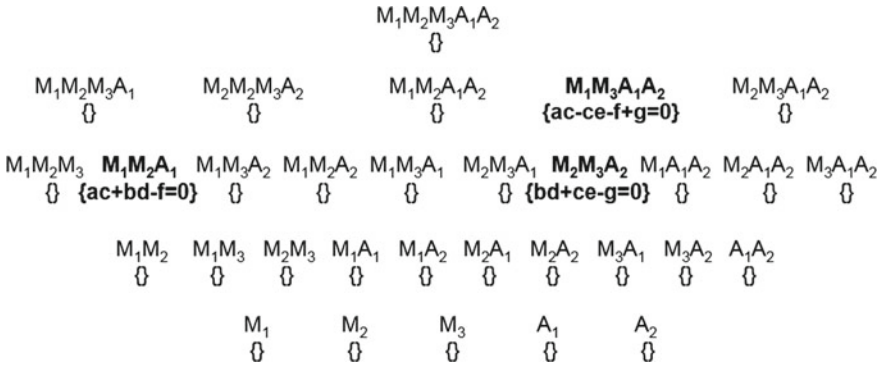


Fig. 14.6 Context network of the polybox example. Relevant contexts are shown in bold

one time in the set of constraints associated with each context because Gröbner Basis algorithm is not able to remove these variables. Finally, the context is relevant, if after the first and second step, there is at least one constraint for each component of the context.

Gröbner Basis algorithm is not applied to irrelevant contexts, because the set of obtained constraints will be empty or included in the set of obtained constraints of relevant contexts. Irrelevant contexts are not necessary for obtaining the complete set of conflicts of a system.

Example In Fig. 14.6, the context network for the polybox example is shown. Gröbner Basis algorithm is applied only to the relevant contexts: $M_2M_3A_1A_2$, $M_1M_2A_2$, and $M_2M_3A_2$.

14.3.5.3 Determination of Minimal Conflict Contexts

A constraint-driven algorithm is proposed in order to obtain the Minimal Conflict Contexts (MCCs), based on the following definition:

Definition 14.10 *Context Analytical Redundancy Constraint (CARC)*. A constraint derived from SM where only observable variables are related. The set of CARCs is the union of all the constraints obtained by applying Gröbner Basis algorithm for each relevant context.

A relevant context RC is an MCC if it has not an empty set of constraints and it satisfies one of the following predicates:

- All its sub-contexts are not MCCs.
- At least one CARC of the context *RC* is not included in any of its sub-contexts which are MCCs.

In order to determine which relevant contexts are MCCs, the graph of relevant contexts is traversed from the leaf nodes to the root, in such a way that only upper relevant contexts that satisfy the definition of MCC will be taken into account.

Example In the polybox example, all relevant contexts are MCCs. There are three CARCs: $ac - ce - f + g = 0$ (context $M_2M_3A_1A_2$), $ac + bd - f = 0$ (context $M_1M_2A_2$), $bd + ce - g = 0$ (context $M_2M_3A_2$).

The set of minimal diagnoses is obtained by applying a standard hitting set algorithm to the MCCs, which are evaluated to false when an observational model is applied. The evaluation of the MCCs and the hitting set algorithm are done in an online process (when an observational model is known), but all previous steps (as the process for obtaining the MCCs) can be done in an offline process and just one time for all possible observational models.

14.4 Application to a Case of Study

In Fig. 14.7, a system of heat exchangers is shown. This system [18] consists of six heat exchangers, three flows f_i come in at different temperatures t_i. There are three different subsystems, each one formed by two exchangers: E1, E2, E3, E4, and E5, E6. Each of the six exchangers and each of the eight nodes of the system are considered as components in order to be able to verify their correct behavior. In [17], the complete SM of the example is shown.

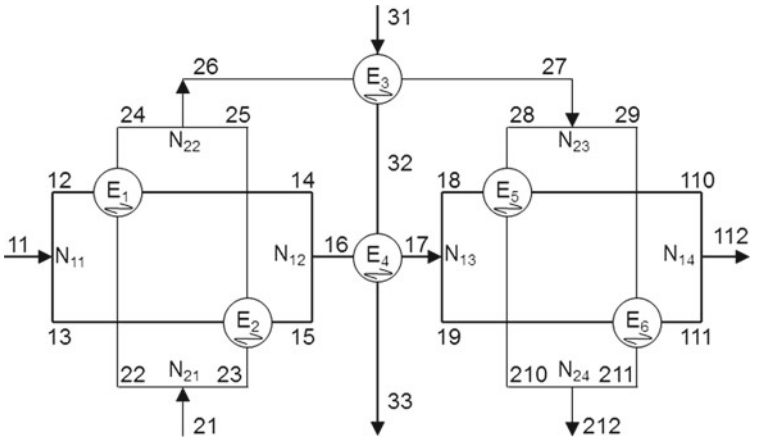
The system of heat exchangers has five clusters: $\{N_{11}\}$, $\{N_{13}\}$, $\{N_{12}, N_{21}, N_{22}, E_1, E_2\}$, $\{N_{14}, N_{23}, N_{24}, E_5, E_6\}$, and $\{E_3, E_4\}$. For example, $\{E_3, E_4\}$ is a cluster because all the connections (inputs and outputs) to other components outside the cluster are observable. There are some non-observable connections inside the cluster, but do not connect to components outside the cluster. For example, the outputs f_{32} and t_{32} of the component E_3 are not observables, but they are inputs for the component E_4.

For each cluster, an independent context network is obtained. This pretreatment reduces the number of nodes of the context network, for example from $2^{14} - 1$ to $(2^1 - 1) + (2^1 - 1) + (2^5 - 1) + (2^5 - 1) + (2^2 - 1) = 67$.

Gröbner Basis algorithm is not applied for irrelevant contexts. For example, context N_{12} is irrelevant because all variables of their constraints are observable. The context $N_{12}N_{21}E_1E_2$ is irrelevant because there exists another context, $N_{12}E_1E_2$, without component N_{21}, that generates the same set of constraints. Another example is E_1E_2, it is an irrelevant context because applying Gröbner Basis is not possible to obtain any constraint.

In Fig. 14.8, the context network for the components cluster $\{N_{12}, N_{21}, N_{22}, E_1, E_2\}$ is shown. Gröbner Basis algorithm is applied only to the relevant contexts: $N_{12}E_1E_2$, $N_{21}N_{22}E_1E_2$, and $N_{12}N_{21}N_{22}E_1E_2$. In Table 14.4, the differences between applying Gröbner Basis algorithm to all contexts (as in [14, 15]) or only to the relevant contexts for each cluster are shown.

In Fig. 14.9, all CARCs grouped by clusters are shown. In order to determine which relevant context is MCC, we have to traverse the graph of the context network



The normal behaviour of the system can be described by means of these constraints:

$\sum_i f_i = 0$: mass balance at each node

$\sum_i f_i \cdot t_i = 0$: thermal balance at each node

$\sum_{in} f_i \cdot t_i - \sum_{out} f_j \cdot t_j = 0$: enthalpy balance for each heat exchanger

C.	Constraints
N_{11}	$\mathbf{f}_{11}$-$\mathbf{f}_{12}$-$\mathbf{f}_{13}$
	$\mathbf{f}_{11}\cdot\mathbf{t}_{11}$-$\mathbf{f}_{12}\cdot\mathbf{t}_{12}$-$\mathbf{f}_{13}\cdot\mathbf{t}_{13}$
N_{12}	$f_{14}+f_{15}$-$\mathbf{f}_{16}$
	$f_{14}\cdot t_{14}+f_{15}\cdot t_{15}$-$\mathbf{f}_{16}\cdot\mathbf{t}_{16}$
N_{13}	$\mathbf{f}_{17}$-$\mathbf{f}_{18}$-$\mathbf{f}_{19}$
	$\mathbf{f}_{17}\cdot\mathbf{t}_{17}$-$\mathbf{f}_{18}\cdot\mathbf{t}_{18}$-$\mathbf{f}_{19}\cdot\mathbf{t}_{19}$
N_{14}	$f_{110}+f_{111}$-$\mathbf{f}_{112}$
	$f_{110}\cdot t_{110}+f_{111}\cdot t_{111}$-$\mathbf{f}_{112}\cdot\mathbf{t}_{112}$
N_{21}	$\mathbf{f}_{21}$-f_{22}-f_{23}
	$\mathbf{f}_{21}\cdot\mathbf{t}_{21}$-$f_{22}\cdot t_{22}$-$f_{23}\cdot t_{23}$

C.	Constraints
N_{22}	$f_{24}+f_{25}$-$\mathbf{f}_{26}$
	$f_{24}\cdot t_{24}+f_{25}\cdot t_{25}$-$\mathbf{f}_{26}\cdot\mathbf{t}_{26}$
N_{23}	$\mathbf{f}_{27}$-f_{28}-f_{29}
	$\mathbf{f}_{27}\cdot\mathbf{t}_{27}$-$f_{28}\cdot t_{28}$-$f_{29}\cdot t_{29}$
N_{24}	$f_{210}+f_{211}$-$\mathbf{f}_{212}$
	$f_{210}\cdot t_{210}+f_{211}\cdot t_{211}$-$\mathbf{f}_{212}\cdot\mathbf{t}_{212}$
E_1	$\mathbf{f}_{12}$-f_{14}
	f_{22}-f_{24}
	$\mathbf{f}_{12}\cdot\mathbf{t}_{12}$-$f_{14}\cdot t_{14}+f_{22}\cdot t_{22}$-$f_{24}\cdot t_{24}$

C.	Constraints
E_2	$\mathbf{f}_{13}$-f_{15}
	f_{23}-f_{25}
	$\mathbf{f}_{13}\cdot\mathbf{t}_{13}$-$f_{15}\cdot t_{15}+f_{23}\cdot t_{23}$-$f_{25}\cdot t_{25}$
E_3	$\mathbf{f}_{26}$-$\mathbf{f}_{27}$
	$\mathbf{f}_{31}$-f_{32}
	$\mathbf{f}_{26}\cdot\mathbf{t}_{26}$-$\mathbf{f}_{27}\cdot\mathbf{t}_{27}+\mathbf{f}_{31}\cdot\mathbf{t}_{31}$-$f_{32}\cdot t_{32}$
E_4	$\mathbf{f}_{16}$-$\mathbf{f}_{17}$
	f_{32}-$\mathbf{f}_{33}$
	$\mathbf{f}_{16}\cdot\mathbf{t}_{16}$-$\mathbf{f}_{17}\cdot\mathbf{t}_{17}+f_{32}\cdot t_{32}$-$\mathbf{f}_{33}\cdot\mathbf{t}_{33}$

C.	Constraints
E_5	$\mathbf{f}_{18}$-f_{110}
	f_{28}-f_{210}
	$\mathbf{f}_{18}\cdot\mathbf{t}_{18}$-$f_{110}\cdot t_{110}+f_{28}\cdot t_{28}$ -$f_{210}\cdot t_{210}$

C.	Constraints
E_6	$\mathbf{f}_{19}$-f_{111}
	f_{29}-f_{211}
	$\mathbf{f}_{19}\cdot\mathbf{t}_{19}$-$f_{111}\cdot t_{111}+f_{29}\cdot t_{29}$ -$f_{211}\cdot t_{211}$

$\mathcal{V}_{observable}$={f_{11},f_{12},f_{13},f_{16},f_{17},f_{18},f_{19},f_{112},f_{23},f_{21},f_{26},f_{27},f_{212},f_{31},f_{33},t_{11},t_{12},t_{13},
t_{16},t_{17},t_{18},t_{19},t_{112},t_{21},t_{26},t_{27},t_{212},t_{31},t_{33}}

$\mathcal{V}_{nonObservable}$={$f_{14}$,$f_{15}$,$f_{21}$,$f_{22}$,$f_{22}$,$f_{23}$,$f_{24}$,$f_{25}$,$f_{28}$,$f_{29}$,$f_{210}$,$f_{211}$,$f_{32}$,$t_{14}$,$t_{15}$,$t_{110}$,$t_{111}$,
t_{22},t_{23},t_{24},t_{25},t_{28},t_{29},t_{210},t_{211},t_{32}}

Fig. 14.7 System of heat exchangers, components, equations, and variables

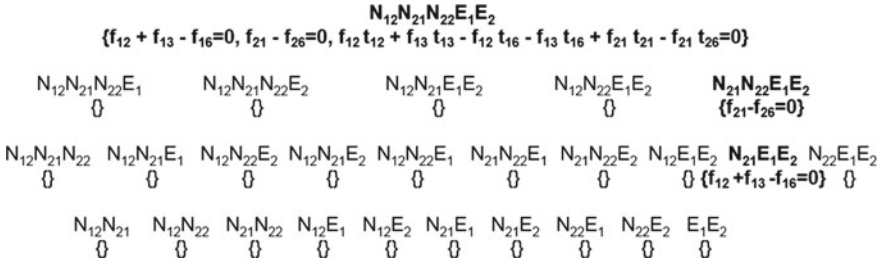


Fig. 14.8 Context network of the cluster $\{N_{12}, N_{21}, N_{22}, E_1, E_2\}$. Relevant contexts are shown in bold

Table 14.4 Improvement by obtaining clusters of components and relevant contexts

	No reduction	Using clusters	Using clusters and Relevant C
Number of nodes of the context net	$2^{14} - 1$	67	67
Calls to Gröbner Basis	$2^{14} - 1$	67	7
Number of obtained constraints	64	14	14

Index	Cluster	Constraints
1	1	$f_{11} - f_{12} - f_{13} = 0$
2	1	$f_{11}\,t_{11} - f_{12}\,t_{12} - f_{13}\,t_{13} = 0$
3	2	$f_{17} - f_{18} - f_{19} = 0$
4	2	$f_{17}\,t_{17} - f_{18}\,t_{18} - f_{19}\,t_{19} = 0$
5	3	$f_{12} + f_{13} - f_{16} = 0$
6	3	$f_{21} - f_{26} = 0$
7	3	$f_{12}\,t_{12} + f_{13}\,t_{13} - f_{12}\,t_{16} - f_{13}\,t_{16} + f_{21}\,t_{21} - f_{21}\,t_{26} = 0$
8	4	$f_{18} + f_{19} - f_{112} = 0$
9	4	$f_{27} - f_{212} = 0$
10	4	$f_{18}\,t_{18} + f_{19}\,t_{19} - f_{18}\,t_{112} - f_{19}\,t_{112} + f_{27}\,t_{27} - f_{27}\,t_{212} = 0$
11	5	$f_{26} - f_{27} = 0$
12	5	$f_{16} - f_{17} = 0$
13	5	$f_{31} - f_{33} = 0$
14	5	$f_{16}\,t_{16} - f_{17}\,t_{17} + f_{26}\,t_{26} - f_{27}\,t_{27} + f_{31}\,t_{31} - f_{31}\,t_{33} = 0$

Fig. 14.9 Context analytical redundancy constraints

from the leaf nodes to the root, in such a way that, if any of the upper contexts do not validate the definition of MCC, it cannot be considered as an MCC. In the heat exchangers example, all the relevant contexts are MCC. In Fig. 14.10, all MCCs of the system are shown.

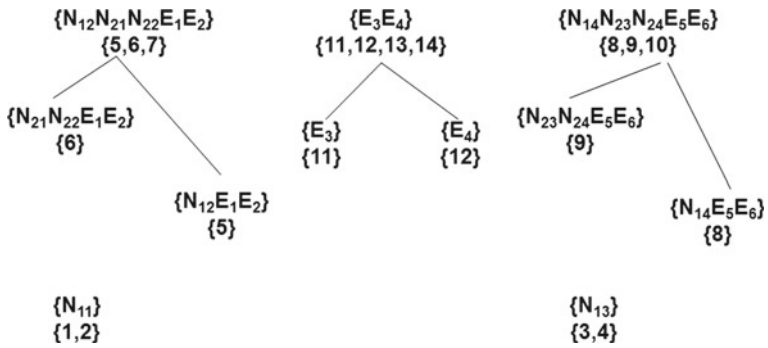


Fig. 14.10 Minimal conflicts contexts network of the system

14.5 Conclusions

The paradigm of Constraint-driven Fault Diagnosis is an adequate alternative to the automation of the fault diagnosis problems for hardware systems. These problems are represented as CSPs/COPs or SAT/MaxSAT in order to detect and identify the possible cause of the faults. The solving process of CSPs/COPs is based on applying consistency and search techniques. The high complexity of the exhaustive search algorithm for certain CSPs/COPs requires the application of decomposition or symbolic techniques in order to reduce this complexity.

Acknowledgements This work has been partially funded by the Ministry of Science and Technology of Spain (TIN2015-63502-C3-2-R) and the European Regional Development Fund (ERDF/FEDER).

References

1. Bakker, R.R., Dikker, F., Tempelman, F., Wogmim, P.M.: Diagnosing and solving overdetermined constraint satisfaction problems. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI'93, vol. 1, pp. 276–281. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
2. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. J. ACM **44**(2), 201–236 (1997)
3. Bose, N.K.: Gröbner bases: an algorithmic method in polynomial ideal theory. In: Multidimensional Systems Theory and Applications, pp. 89–127. Springer Netherlands, Dordrecht (1995)
4. Ceballos, R., Gómez-López, M.T., Gasca, R.M., Del Valle, C.: Determination of possible minimal conflict sets using components clusters and grobner bases. In: Proceedings of the 15th International Workshop on Principles of Diagnosis, DX04, Carcassonne, France, pp. 21–26 (2004)
5. Cordier, M., Dague, P., Dumas, M., Lévy, F., Montmain, J., Staroswiecki, M., Travé-Massuyès, L.: A comparative analysis of AI and control theory approaches to model-based diagnosis. In:

ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, pp. 136–140, 20–25 Aug 2000
6. Davis, R.: Diagnostic reasoning based on structure and behavior. Artif. Intell. **24**(1), 347–410 (1984)
7. Dechter, R., Pearl, J.: The anatomy of easy problems: a constraint-satisfaction formulation. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, CA, USA, pp. 1066–1072, Aug 1985
8. Donald, B.R., Kapur, D., Mundy, J.L.: Symbolic and numerical computation for artificial intelligence (1992)
9. El Fattah, Y., Dechter, R.: Diagnosing tree-decomposable circuits. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI'95, pp. 1742–1749. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995)
10. Freuder, E., Mackworth, A.: Constraint-Based Reasoning. A Bradford Book. MIT Press (1994)
11. Freuder, E., Wallace, R.: Partial constraint satisfaction. Artif. Intell. **58**, 21–70 (1992)
12. Freuder, E.C.: Synthesizing constraint expressions. Commun. ACM **21**(11), 958–966 (1978)
13. Gasca, R.M., Del Valle, C., Gómez-López, M.T., Ceballos, R.: NMUS: structural analysis for improving the derivation of all MUSes in overconstrained numeric CSPs. In: Borrajo, D., Castillo, L., Corchado, J.M. (eds.) Current Topics in Artificial Intelligence, pp. 160–169. Springer, Berlin, Heidelberg (2007)
14. Gasca, R.M., Ortega, J., Toro, M.: Diagnosis dirigida por restricciones simbólicas para modelos polinómicos. In: Diagnosis, Razonamiento Cualitativo y Sistemas Socieconómicos, pp. 71–78. Carlos Alonso y J. A. Ortega (2001)
15. Gasca, R.M., Ortega, J.A., Toro, M.: Diagnosis basada en modelos polinomicos usando técnicas simbólicas. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial **5**(14), 68–77 (2001)
16. Genesereth, M.R.: The use of design descriptions in automated diagnosis. Artif. Intell. **24**(1–3), 411–436 (1984)
17. Gómez-López, M.T., Ceballos, R., Gasca, R.M., Del Valle, C.: Applying constraint databases in the determination of potential minimal conflicts to polynomial model-based diagnosis. In: Constraint Databases, Proceedings of the 1st International Symposium on Applications of Constraint Databases, CDB'04, Paris, pp. 75–89, 12–13 June 2004
18. Guernez, C.: Fault detection and isolation on non linear polynomial systems. In: 15th IMACS World Congress on Scientific, Computation, Modelling and Applied Mathematics (1997)
19. Hoffmann, C.M.: Geometric and Solid Modeling: An Introduction. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1989)
20. Junker, U.: QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In: Proceedings of the 19th National Conference on Artifical Intelligence, AAAI'04, pp. 167–172. AAAI Press (2004)
21. de Kleer, J.: An assumption-based TMS. Artif. Intell. **28**(2), 127–162 (1986)
22. de Kleer, J., Mackworth, A.K., Reiter, R.: Characterizing diagnoses and systems. Artif. Intell. **56**(23), 197–222 (1992)
23. Kumar, V.: Algorithms for constraint satisfaction problems: a survey. AI Mag. **13**(1), 32–44 (1992)
24. Liffiton, M.H., Moffit, M.D., Pollack, M.E., Sakallah, K.A.: Identifying conflicts in overconstrained temporal problems. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI'05, pp. 199–204 (2005)
25. Liffiton, M.H., Sakallah, K.A.: On finding all minimally unsatisfiable subformulas. In: Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing, SAT 2005. Lecture Notes in Computer Science. Springer (2005)
26. Sachenbacher, M., Williams, B.: Diagnosis as semiring-based constraint optimization. In: Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'04, pp. 873–877. IOS Press, Amsterdam (2004)
27. Stumptner, M., Wotawa, F.: Diagnosing tree-structured systems. Artif. Intell. **127**(1), 1–29 (2001)