

# Designing testable subsystems using relational algebra and structural modeling

Abed Alrahim Yassine\* Stéphane Ploix\* Jean-Marie Flaus\*

\* *G-SCOP, Laboratoire des Sciences pour la Conception,  
l'Optimisation et la Production de Grenoble - UMR5272, France  
(e-mail: first name.family name@g-scop.inpg.fr)*

---

**Abstract:** In complex industrial plants, there are usually many sensors and the modeling of plants leads to lots of mathematical relations. This paper presents a general method for finding all the possible testable subsystems i.e. sets of relations than can lead to various types of detection tests. This method which combines relation algebra with structural modeling provides efficiently the constraints that have to be used for the design of each detection test. Comparing to existing approaches, the resulting algorithm is very flexible. Indeed, it is able to manage situations where constraints contain non deductible variables and where some constraints cannot be gathered in the same test. Thanks to these results, it becomes possible to select the most interesting testable subsystems regarding detectability and diagnosability criteria. A example is given in order to point out the capabilities of the method compared to existing methods.

---

## 1. INTRODUCTION

Generally speaking, diagnostic analysis of physical systems rely on detection tests that provide symptoms. In the scientific literature, there are two main trends for the design of tests. The first one comes from the Artificial Intelligence community (Reiter [1987], De Kleer and Williams [1987], Dague [2001], Pulido and Alonso [2002]). It relies on component-based approaches. The principle is to model the different components with relations, also named constraints, and to directly check the consistency between the models and the data. The second stream coming from the Fault Detection and Isolation community (Patton et al. [1989], Frank [1990], Willsky [1976]) uses a complete model, usually a state space representation, of the system to be diagnosed. It is often called a structured or robust approach because it aims at projecting observations in different subspaces in order to distinguish the different faults that may occur (and to remove uncertain parts). These approaches raise another issue: projections do not trace the components that are involved into the tests. It is therefore difficult to interpret the symptoms, especially with formal analysis such as in (Nyberg and Krysander [2003], Ploix et al. [2003]).

Complex industrial systems can be depicted by various kinds of relations, also named constraints, depending on the modeling approach: mathematical equations, qualitative relations, rules expressed in a natural language, . . . may be encountered. Methods that can manage both complexity and models of different types are necessary. This fact necessitates the use of structural approaches and rules out approaches based on state space representations or on Grobner bases (Frisk [2000]). In diagnosis, structural modeling has been introduced in (Davis [1984]). Using a semantic theory of abstraction (Nayak and Levy [1995], Chittaro and Ranon [2004]) has pointed out that structural modeling is an abstraction of behavioral modeling. Then, discovering testable sets of constraints can be

achieved thanks to a procedure based on a structural model such as the bipartite graph approach (Dulmage and Mendelsohn [1959]) proposed in (Blanke et al. [2006], Declerck and Staroswiecki [1991]). Finding testable sets of constraints may indeed be done thanks to an elimination procedure that combines constraints in order to eliminate all the unknown physical variables and therefore getting constraint containing only known data i.e. testable constraints. In (Travé-Massuyès et al. [2006b]), an alternative method has been proposed but the level of complexity is still high. (Krysander et al. [2005]) has proposed an improved algorithm but it does not manage constraints containing non-deductible variables.

This paper presents a general method that provides testable subsystems. It uses a join-operator coming from relational algebra to model value propagations or variable elimination between constraints. Thanks to this operator which abstracts propagations, testable subsystems can be found using less intuitive but more efficient ways. A algorithm, which improves (Ploix et al. [2005]) and is more general than (Krysander et al. [2005]), based on a join-operator coming from relational algebra is then proposed. It relies also on a structural abstraction of the constraints and it traces all the constraints that are involved in testable subsystems thus making it possible to determine what physical components are examined by each test. An example is presented in order to point out the performance of the algorithm.

## 2. RELATIONAL ALGEBRA AND JOIN-OPERATOR

Relational Algebra (RA) can be viewed as a data manipulation language for relational model. It is mainly used for the modeling of relational databases but this paper proposes an extension of RA to combine relations structurally modelled in order to design ARR. RA consists of several basic operations which make it possible to specify data

requests. We can distinguish three families of relational operators:

- unary operators (Project  $\Pi$ , Select  $\sigma$ )
- set-membership binary operators (Union  $\cup$ ), Intersect  $\cap$ , Difference  $-$ )
- binary operators (Cartesian product  $\times$ ), join  $\bowtie$ )

A relationship  $R$  is usually represented by a class or by a table. It is denoted:  $R(A_1, A_2, \dots, A_n)$  such as  $A_1, A_2, \dots, A_n$  represent the attributes of the relationship  $R$ .  $R$  may also a mathematical relation where attributes correspond to variables. In the next, the cartesian product and the join operator are presented because they are useful for our purpose.

*Definition 1.* (selection) A selection, also called restriction, leads to a set of tuples belonging to a relation  $R(A_1, A_2, \dots, A_n)$  that satisfies a logical condition defined over  $A_1 \times A_2 \times \dots \times A_n$ . It is denoted  $\sigma_E R$ .

Selection is useful to link different relations: it is useful to impose the same value for a variable appearing in different relations: it corresponds to a value propagation between constraints.

*Definition 2.* (projection) A projection of a relation  $R(A_1, A_2, \dots, A_n)$  according to some attributes  $A_1, \dots, A_m$ ;  $m < n$  amounts to keep the tuples corresponding to attributes  $A_{m+1}, \dots, A_n$  and to remove duplicated resulting tuples.

Projection is useful to remove variables.

*Definition 3.* (cartesian product) The Cartesian product of two relationships  $R_1$  and  $R_2$  is the set of all possible ordered pairs whose first tuple belongs to  $R_1$  and whose second tuple belongs to  $R_2$ . It is denoted:  $R_1 \times R_2 = \{(r_i, r_j) : r_i \in R_1 \text{ and } r_j \in R_2\}$

Cartesian product is useful to combine relations.

*Definition 4.* (Join) The join operator is a operator defined in the relational data model (Codd [1970], Mishra and Eich [1992]). It is used to build a new relationship  $R$  by combining two relationships  $R_1$  and  $R_2$ . This relationship  $R$  includes all possibilities of combining pairs of tuples coming respectively from relationships  $R_1$  and  $R_2$ , which satisfy a join condition  $E$

Indeed, the join operator is a cartesian product followed by a select:  $R_1 \bowtie_E R_2 = \sigma_E(R_1 \times R_2)$ .

*Definition 5.* (theta-join) A theta-join is a join such as the join condition  $E$  is a simple comparison between a attribute  $A_1$  of the relationship  $R_1$  and a attribute  $A_2$  of the relationship  $R_2$ . It is denoted  $R_1 \bowtie_E R_2$ . The join condition can be one of the following:  $=, \neq, \leq, \geq, <, >$ .

*Definition 6.* (equi-join) A equi-join is a theta-join such as the condition join  $E$  is a equality between an attribute  $A_1$  of the relationship  $R_1$  and an attribute  $A_2$  of the relationship  $R_2$ . The equijoin is denoted:  $R_1 \bowtie_{A_1=A_2} R_2$ . Indeed, the equi-join is a cartesian product followed by a select:  $R_1 \bowtie_{A_1=A_2} R_2 = \sigma_{A_1=A_2}(R_1 \times R_2)$

The equi-join operator is useful to model a value propagation between two constraints or, equivalently, elimination of a variables between two constraints.

The concept of value propagation between two constraints  $k_i$  and  $k_j$  is similar to the projection of the equi-join of the two constraints.

*Definition 7.* (propagation) The propagation of a value  $v$  between two constraints  $k_i$  and  $k_j$  may be represented by the join operator:

$k_i \bowtie_v k_j = \Pi_{(var(k_i) \cup var(k_j) \setminus \{v\})}(k_i \bowtie_{k_i.v=k_j.v} k_j)$  such as  $var(k_i)$  and  $var(k_j)$  represent the sets of variables appearing in the constraints  $k_i$  and  $k_j$ .

The join operator  $k_i \bowtie_v k_j$  is used as a basic concept for the design of the testable subsystems of a system.

### 3. PROBLEM STATEMENT

The behavior of a system is described with a model. Usually the model is a set of constraints  $K$ . A structural model contains the information of which variables that are contained in each constraint.

The variables of a constraint  $k$  denoted  $V = var(k)$  can be broken down into a set of deductible variables and a set of non-deductible variables. A structure  $s$  will be written  $\lfloor V^-, V^+ \rfloor$  where  $V^- = var^-(s)$  represents the non-deductible variables of  $s$  and  $V^+ = var^+(s)$  represents the deductible variables of  $s$ .  $V^-$  and  $V^+$  satisfy  $V^- \cap V^+ = \emptyset$ . Therefore, the structure modeling a constraint  $k$  is denoted:  $s(k) = \lfloor V^-, V^+ \rfloor$ .

For the sake of simplicity, the following notations have been adopted:  $\lfloor \emptyset, V^+ \rfloor = \lfloor V^+ \rfloor$  and, if  $S$  is a set of structures,  $var(S) = \bigcup_{s \in S} var(s)$ . Finally, an empty structure  $s$  is a structure satisfying:  $var(s) = \emptyset$ . It is denoted:  $s = \lfloor \rfloor$ .

For the design of testable subsystem, some structures are particularly useful because they model what it is known in a system i.e. the controlled or measured variables: they are named terminal structure.

*Definition 8.* A terminal structure  $s$  satisfies  $|var(s)| = 1$ . It usually involves a data flow and models the fact that a trajectory can be assigned to a variable. By extension, a constraint containing only one variable is also qualified as terminal.

Because of the possible over-estimations, it is useful to introduce the following definition.

*Definition 9.* A structure  $s_1$  wraps another structure  $s_2$  if  $var(s_1) \supseteq var(s_2)$  and  $var^+(s_1) \supseteq var^+(s_2)$ . It is denoted:  $s_1 \supseteq s_2$ .

In order to show that values can be propagated between constraints, i.e. that the intersection of two constraints can be projected without loss, a join operator is defined. As a prerequisite, the notion of propagable variable has to be introduced.

*Definition 10.* Let  $s_1$  and  $s_2$  be two structures related to constraints  $k_1$  and  $k_2$ . The propagation of a variable  $v$  from  $s_1 = s_1(k_1)$  to  $s_2 = s_2(k_2)$  is possible only if  $v \in var^+(s_1)$  and if  $v \in var(s_2)$ . The variable  $v$  is qualified as propagable between  $s_1$  and  $s_2$  if it is propagable from  $s_1$  to  $s_2$  or propagable from  $s_2$  to  $s_1$ .

The join operator can now be defined.

*Definition 11.* Let  $s_1$  and  $s_2$  be two structures, with  $V_1^+ = \text{var}^+(s_1)$ ,  $V_1^- = \text{var}^-(s_1)$ ,  $V_2^+ = \text{var}^+(s_2)$  and  $V_2^- = \text{var}^-(s_2)$ . The join operator, denoted  $\bowtie_v$ , where  $v$  is a propagable variable between  $s_1$  and  $s_2$ , is defined only in the following three situations:

- if  $\{v\} \in V_1^+ \cap V_2^-$  then

$$s_1 \bowtie_v s_2 = \perp (V_1^- \cup V_1^+ \cup V_2^-) \setminus (V_2^+ \cup \{v\}), V_2^+ \perp$$

- if  $\{v\} \in V_2^+ \cap V_1^-$  then

$$s_1 \bowtie_v s_2 = \perp (V_1^- \cup V_2^+ \cup V_2^-) \setminus (V_2^+ \cup \{v\}), V_2^+ \perp$$

- if  $\{v\} \in V_1^+ \cap V_2^+$ , then

$$s_1 \bowtie_v s_2 = \perp (V_1^- \cup V_2^-) \setminus (V_1^+ \cup V_2^+), (V_1^+ \cup V_2^+) \setminus \{v\} \perp$$

If a formula  $s_1 \bowtie_v s_2$  satisfies one of the three previous points, it is qualified as evaluable.

Using this operator results in a definition of a propagation method i.e. if the value of a variable can be deduced from one constraint, then this value can be propagated into the other one. A link is thus created between two constraints: it corresponds partially to the join operator in relational algebra.

*Theorem 1.* Let  $k_1$  and  $k_2$  be two constraints. Then a wrapping structure of the constraint resulting from the propagation of a variable  $v$  between  $k_1$  and  $k_2$  can be obtained using  $\bowtie_v$  operator on  $s(k_1)$  and  $s(k_2)$ .

The proof is presented in (Ploix et al. [2008]).

This section introduced the structures of constraints and a join operator that models value propagations between structures. These tools can then be used to design testable subsystems containing many different kinds of constraints. The way to use these tools is described in the next section.

#### 4. TESTABLE SUBSYSTEM DESIGN

Some basic concepts have first to be introduced before tackling the design of testable subsystems.

##### 4.1 Combining structures into formulae

A test results from consecutive propagations that can be modeled by a propagation formula:  $f = (((s_1 \bowtie_{v_1} s_2) \bowtie_{v_2} s_3) \bowtie_{v_3} (s_4 \bowtie_{v_4} s_2)) \dots$ . A formula is composed of subformulae linked to the join operator, where the elementary formulae are structures. Thanks to this operator, a propagation formula  $f$  can be evaluated as a structure  $s(f)$ . The set of all the formulae is denoted  $\mathbb{F}$ .

*Definition 12.* A formula is qualified as evaluable if all its subformulae and itself are evaluable.

*Definition 13.* The support of a formula  $f \in \mathcal{F}$ , denoted  $\sigma(f)$ , is the set of all the structures involved in the formula.

*Definition 14.* The degree of a formula  $f \in \mathbb{F}$ , denoted  $d(f)$ , is equal to the number of times the join operator appears in the formula: it represents the number of elementary propagations.

*Definition 15.* Two formulae  $f_1$  and  $f_2$  are comparable if  $\sigma(f_1) = \sigma(f_2)$  and if  $\text{var}(s(f_1)) = \text{var}(s(f_2))$  i.e. they have the same constraints and the same variables. This is denoted:  $f_1 \sim f_2$ .

Moreover, during propagations, a given variable can be instantiated only once. However, in some formulae, a variable can appear several times and then be instantiated in different ways. In this situation, there will be a simpler formula where the variable has been instantiated only once.

*Definition 16.* A formula  $f_1$  is simpler than a formula  $f_2$  if:

- $\sigma(f_1) \subset \sigma(f_2)$  and  $\text{var}(s(f_1)) \subset \text{var}(s(f_2))$
- or if  $f_1 \sim f_2$  and  $d(f_1) < d(f_2)$
- or if  $f_1 \sim f_2$  and  $d(f_1) = d(f_2)$  and  $\text{var}^+(s(f_1)) \subset \text{var}^+(s(f_2))$

It is denoted:  $f_1 < f_2$ . It is said that  $f_2$  overestimates  $f_1$ .

A testable propagation formula  $f \in \mathbb{F}$  is an evaluable formula that leads to an empty structure. A testable propagation formula is minimal over a set of structures  $S$  if there is no simpler formula over  $S$ . It is called the minimal testable propagation formula (MTPF) over  $S$ .

According to the definition of the join operator, all the formulae correspond to wrapping structures of constraints that can be found by combining elementary constraints  $K = \{\dots, k_i, \dots\}$  of a diagnostic problem. Therefore, the formulae may over-estimate the propagations that lead to a test. Fortunately, it is easy to check if all the constraints related to the support  $\sigma(f)$  of an MTPF  $f$  have been used during the design of a test.

##### 4.2 Characteristic of the combining procedure

Applying the join operator  $\bowtie_v$  on a variable  $v$  belonging to two formulae  $f_i$  and  $f_j$  yields a new formula:  $f_i \bowtie_v f_j$  while removing  $f_i$  and  $f_j$ . The number of formulae is therefore reduced by one. Applying the join operator  $\bowtie_v$  on a variable  $v$  belonging to three formulae  $f_i$ ,  $f_j$  and  $f_k$  yields three new formulae:  $f_i \bowtie_v f_j$ ,  $f_i \bowtie_v f_k$  and  $f_j \bowtie_v f_k$ . The number of formulae remains constant. Applying the join operator  $\bowtie_v$  on a variable  $v$  belonging to four formulae  $f_i$ ,  $f_j$ ,  $f_k$  and  $f_l$  yields six new formulae:  $f_i \bowtie_v f_j$ ,  $f_i \bowtie_v f_k$ ,  $f_i \bowtie_v f_l$ ,  $f_j \bowtie_v f_k$ ,  $f_j \bowtie_v f_l$  and  $f_k \bowtie_v f_l$ . Figure 1 shows the link between the number of formulae before and after applying the join operator. It points out that it is better to start applying the join operator to variables with lowest order to keep low the number of formulae. The resulting number of formulae depends on the number of occurrences of variables within the structures corresponding to a set of formulae. This number is called the order of a variable. Let  $F$  be a set of formulae. The number of formulae from  $F$  where a variable  $v$  appears in the related structures, is named the order of  $v$  in  $F$ . It is denoted:  $o_F(v)$ .

Next section presents an algorithm that uses the characteristics to reduce the complexity of the design of testable subsystems.

##### 4.3 Algorithms for the design of testable subsystems

Some definitions and lemmas presented in Yassine et al. [2008] are recalled to be able to remove useless structures.

*Definition 17.* A set of constraints  $K \subset K_\Sigma$  is linked in  $K_\Sigma$  by a set of variables  $V \subset \text{var}(K_\Sigma)$  iff  $K$  there is a complete matching with respect to the variables

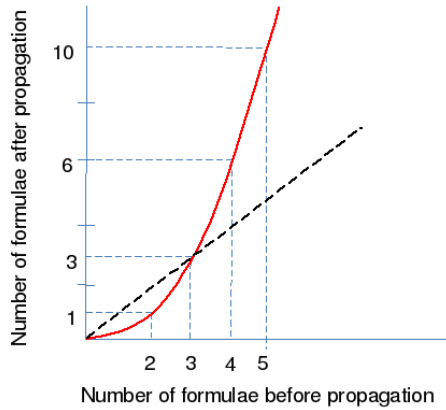


Fig. 1. Influence of the join operator on the number of formulae

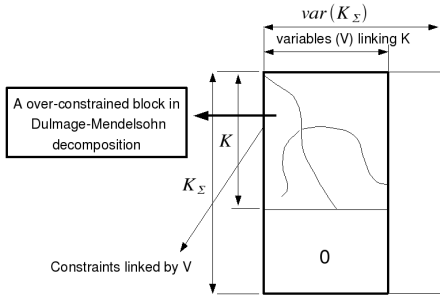


Fig. 2. Structural matrix of a constraint set  $K$ , which is linked by a set of variables  $V$

$V$  (see Dustegor et al. [2006]) and there is only one constraint which does not belong to the matching and  $K_{\Sigma} \setminus K$  do not contain any variable of  $V$ . The variables of  $V$  are called linking variables for  $K$ . They are denoted:  $var_{linking}(K, K_{\Sigma})$ .

The shape of a structural matrix dealing with linked constraints is drawn in figure 2.

*Definition 18.* A set of several constraints  $K \subset K_{\Sigma}$  is isolated in  $K_{\Sigma}$  by a set of variables  $V \subset var(K_{\Sigma})$  if it is linked by  $V$  and if there is at least one variable in  $var(K) \setminus V$  that does not belong to other constraints of  $K_{\Sigma}$  (i.e.  $K_{\Sigma} \setminus K$ ). If the set contains only one constraint, the link condition disappears.

The shape of a structural matrix dealing with isolated constraints is drawn in figure 3.

*Lemma 2.* A sufficient condition for a subset of constraints  $K \subset K_{\Sigma}$  to be non detectable (it cannot be included in a TSS) is that there is a tuple  $(K_1, \dots, K_m)$  of  $m$  sets of constraints making up a partition  $\mathcal{P}(K)$  of  $K$  such that each  $K_i$  is isolated in  $K_{\Sigma} \setminus \bigcup_{j < i} K_j$  ( $K_1$  is a limit case: it should be isolated in  $K_{\Sigma}$ ).

This tuple corresponds to several just-determined blocks or under-determined blocks in Dulmage-Menlesohn decomposition.

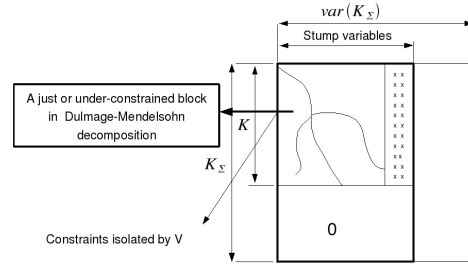


Fig. 3. Structural matrix of a constraint set, which is isolated by the set of variables  $V$

The proofs of these definitions and lemmas are presented in (Yassine et al. [2008]).

For the design of testable subsystems (TSS), all the possible MTPF have to be found because TSS are given by the support of MTPF. The principle is to iteratively propagate values until MTPF are found. But, unlike value propagation, a propagation can be considered even if related variables have not been instantiated. In order to reduce the computations, the propagations related to a variable that involve the fewest structures are achieved first.

Sets of formulae are represented by the letter  $F$  and the corresponding structures by  $s(F) = \{s(f); f \in F\}$ .

Consider the set of constraints intervening in a diagnostic problem and  $F_0$ , the corresponding formulae, which are also elementary formulae.  $s(F_0)$  denotes the structures corresponding to  $F_0$ .

Before starting the computation of MTPF, it is better to remove formulae that cannot appear in a TSS (see lemma 2). These formulae can be found using the Dulmage Mendelsohn approach proposed in (Cassar and Staroswiecki [1997]) on the related structures. The formulae corresponding to structures appearing in under and just determined blocks have to be removed.

The resulting cleared set of formulae is named  $F_1$ . The propagations can now be achieved according to the orders of variables. The variables of lowest order are selected first. Let  $v$  be one of these variables. All the formulae where  $v$  appears are selected and, using the join operator, new evaluable formulae are then deduced and added to the current set of formulae. Formulae that overestimate others are removed at each step as well as formulae that contain  $v$ . This procedure is repeated until all the variables have been considered. Remaining structures are then empty structures and thus all the MTPF have been found. The procedure is summarized by the following algorithm.

Sometimes, because the number of testable propagation formulae is very high, it is quicker to find only a subset of all the MTPF. In order to reduce to number of propagations but also to check all the constraints, propagations of variables that are known because they have been measured or controlled, can be reduced: propagations may indeed be stopped when a known variable is found. The resulting MTPF are called basic MTPF (the basic MTPFs are equivalent to the basic TSSs in Travé-Massuyès et al. [2006a]). They are valuable because basic MTPF are a small subset of all the MTPF that covers all the exploitable

---

**Algorithm 1** Compute MTPF
 

---

**Require:**  $F_1$ , a set of formulae

 $F \leftarrow F_1$ 
**while**  $\text{var}(s(F)) \neq \emptyset$  **do**
**select**  $v \in \text{var}(s(F))$  **such that**  $o_F(v) \leq o_F(v_i), \forall v_i \in \text{var}(s(F))$ 
 $F' \leftarrow \{f/v \in \text{var}(s(f))\}$ 
 $F'' \leftarrow \{f_i \bowtie_v f_j; (f_i, f_j) \in F'^2, i \neq j, f_i \bowtie_v f_j \text{ evaluable}\}$ 
 $F \leftarrow (F \setminus F') \cup F''$ 
 $F \leftarrow F \setminus \{f \in F; \exists f_i \in F, f_i \neq f, f \succ f_i\}$ 
**end while**
**return**  $F$ 


---

structures present in all the MTPF. Algorithm 1 can still be used but when a variable  $v$  is involved in a terminal structure, the join operator  $\bowtie_v$  is only applied, on the one hand, between the terminal structures and the other structures involving  $v$ , and on the other hand, between the terminal structures themselves if many of them include the variable  $v$ : it corresponds to the so-called material redundancy.

Using the join operator according to the order of variables is an improvement regarding complexity (see figure 1). The join operator can also be applied in other ways in order to provide other formulae leading to testable subsystems. For instance, it can also be used to propagate variables according to a complete matching wrt variables (see Cassar and Staroswiecki [1997]) but, in this case, properties illustrated in figure 1 would not be exploited. Because the join operator makes propagation or elimination more abstract, it opens new perspectives for improvements of method for the design of testable subsystems.

## 5. COMPARISON WITH OTHER APPROACHES

Other approaches based on bipartite graphs and Dulmage-Mendelsohn decomposition are compared to the proposed solution. Because the complexity of algorithms is highly dependent on the problem to be solved and because upper bounds for complexity are not meaningful, the different approaches already in use have been compared with the proposed one using an example. The example has been chosen without non-deductible variables because deductibility has not been taken into account in existing approaches. Consider a system composed by 4 iterations of a electronic counter equipped with a digital display. The system is modeled by the following constraints:

$$\begin{aligned} k_1 : x_1 &= x_0 + 1 & k_6 : \tilde{x}_1 &= x_1 \\ k_2 : x_2 &= x_1 + 1 & k_7 : \tilde{x}_2 &= x_2 \\ k_3 : x_3 &= x_2 + 1 & k_8 : \tilde{x}_3 &= x_3 \\ k_4 : x_4 &= x_3 + 1 & k_9 : \tilde{x}_4 &= x_4 \\ k_5 : \tilde{x}_0 &= x_0 & & \end{aligned}$$

### 5.1 Bipartite graph approach

The bipartite graph approach (Dulmage and Mendelsohn [1959]) has been used in (Blanke et al. [2006], Declerck and Staroswiecki [1991]) to compute the analytical redundancy relations (ARR): it corresponds to the supports of MTPF. The bipartite graph approach is a structural approach, which only requires knowing the variables that occur

in constraints. The bipartite graph is defined by  $G = (K, \text{var}(K), \mathcal{E})$  where  $K$  is the constraints set and  $\mathcal{E} = K \times \text{var}(K)$  stands for edges in the graph. Complete matchings within a connected graph i.e. a sub-graph containing all the variables  $\text{var}(K)$  and whose edges contain no common vertex, neither constraint nor variable, are then selected in order to find all the ARR.

A matching directs the search of ARR. ARR are composed of alternated chains starting with a terminal constraints and ending with a unmatched constraint. In a connected bipartite graph, for a given matching, the number of ARR that can be found is equal to the number of unmatched sets of equivalent realizations. From these constraints, the bipartite graph in figure 4 can be drawn up.

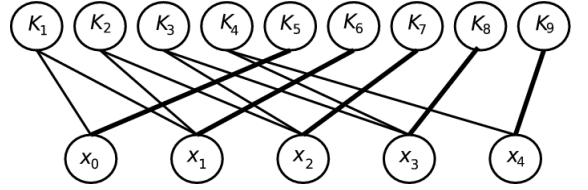


Fig. 4. Bipartite graph with a matching for the counter

The following matching can firstly be chosen:

$\{(k_5, x_0), (k_6, x_1), (k_7, x_2), (k_8, x_3), (k_9, x_4)\}$ . By applying the approach proposed in (Blanke et al. [2006]), 4 testable subsystems are obtained:  $\{k_1, k_2, k_5, k_7\}$ ,  $\{k_2, k_6, k_7\}$ ,  $\{k_3, k_7, k_8\}$  and  $\{k_4, k_8, k_9\}$ .

These testable subsystems lead to the 4 ARR:  $ARR_1 : \tilde{x}_0 - \tilde{x}_1 + 1 = 0$ ,  $ARR_2 : \tilde{x}_1 - \tilde{x}_2 + 1 = 0$ ,  $ARR_3 : \tilde{x}_2 - \tilde{x}_3 + 1 = 0$  and  $ARR_4 : \tilde{x}_3 - \tilde{x}_4 + 1 = 0$ . Nevertheless, there are obviously 10 ARR in this example. In order to obtain other ARR, other matchings have to be considered. For instance, the matching  $\{(k_1, x_0), (k_6, x_1), (k_7, x_2), (k_8, x_3), (k_9, x_4)\}$  leads to new ARR.

This example shows that a matching makes it possible to obtain only one part of the set of all the ARR. The whole set can be obtained provided that all the possible matchings have been considered. In this example, 62 possible matchings can be found. For each matching, the number of ARR that can be found is equal to the number of constraints not matched (4 in this example). Therefore, in order to finally keep only 10 ARR,  $64 \times 4$  redundant ARR are obtained in this simple example.

### 5.2 Krysanter's approach

(Krysanter et al. [2008]) proposed a algorithm for finding all **MSOs** (minimal structurally over-constrained subsystems), i.e testable subsystems. This algorithm is based on the Dulmage-Mendelshon decomposition (Dulmage and Mendelsohn [1959]) and on a top-down approach. It starts with the entire model and then reduce the size of the model step by step until an TSS remains.

To understand this algorithm, the notion of structural redundancy must be presented.

Given bipartite graph  $G = (K, \text{var}(K), \mathcal{E})$ . The structural redundancy  $\varphi K$  is defined by:

$$\varphi K = |K^+| - |\text{var}(K^+)|.$$

where  $K^+$  is the structurally overdetermined part of  $K$ .

This part is equal to the vertical tail of the Dulmage-Mendelshon decomposition of  $G$ . In a **TSS**, the structural redundancy is 1.

The digital system will be used to illustrate this method. This system may be presented by the structural matrix

Table 1. the structural matrix of system

|       | $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|-------|-------|-------|-------|-------|-------|
| $k_1$ | 1     | 1     | 0     | 0     | 0     |
| $k_2$ | 0     | 1     | 1     | 0     | 0     |
| $k_3$ | 0     | 0     | 1     | 1     | 0     |
| $k_4$ | 0     | 0     | 0     | 1     | 1     |
| $k_5$ | 1     | 0     | 0     | 0     | 0     |
| $k_6$ | 0     | 1     | 0     | 0     | 0     |
| $k_7$ | 0     | 0     | 1     | 0     | 0     |
| $k_8$ | 0     | 0     | 0     | 1     | 0     |
| $k_9$ | 0     | 0     | 0     | 0     | 1     |

The structural redundancy of the system is 4.

In order to find the TSSs, The algorithm proposed in (Krysander et al. [2008]) is applied:

- If  $k_1$  is removed, the Dulmage-Mendelshon decomposition of  $K \setminus \{k_1\}$  is  $(K \setminus \{k_1\})^+ = \{k_2, k_3, k_4, k_6, k_7, k_8\}$ ,  $(K \setminus \{k_1\})^0 = \{k_5\}$  and  $(K \setminus \{k_1\})^- = \{\emptyset\}$ . The set  $K' = (K \setminus \{k_1\})^+$  does not verifies  $\varphi K' = 1$ , then the set  $K'$  is not a TSS.
- If  $k_1, k_2$  are removed, the Dulmage-Mendelshon decomposition of  $K \setminus \{k_1, k_2\}$  is  $(K \setminus \{k_1, k_2\})^+ = \{k_3, k_4, k_7, k_8\}$ ,  $(K \setminus \{k_1, k_2\})^0 = \{k_5, k_6\}$  and  $(K \setminus \{k_1, k_2\})^- = \{\emptyset\}$ . The set  $K' = (K \setminus \{k_1, k_2\})^+$  does not verifies  $\varphi K' = 1$ , then the set  $K'$  is not a TSS.
- If  $k_1, k_2, k_3$  are removed, the Dulmage-Mendelshon decomposition of  $K \setminus \{k_1, k_2, k_3\}$  is  $(K \setminus \{k_1, k_2, k_3\})^+ = \{k_4, k_8, k_9\}$ ,  $(K \setminus \{k_1, k_2, k_3\})^0 = \{k_5, k_6, k_7\}$  and  $(K \setminus \{k_1, k_2, k_3\})^- = \{\emptyset\}$ . The set  $K' = (K \setminus \{k_1, k_2, k_3\})^+$  verifies  $\varphi K' = 1$ , then the set  $K' = \{k_4, k_8, k_9\}$  is a TSS.

In order to find all the testable subsystems, all the combinations of elimination must to be made. In this example, 119 combination have been achieved in order to find the 10 possible **TSSs** of the system.

The same TSS can be found more than once. So, this algorithm is not optimal in terms of efficiency. (Krysander et al. [2008]) presented improvements in order to increase the efficiency.

Even if the complexity of this method is lower than for the last approach, the main disadvantage of this method is that it can not take into account the notion of deductibility of variables: all variables of the system are considered as deductibles and, therefore, some **ARRs** may not be achievable. Of course, systems with branchings cannot be managed.

### 5.3 Assessment of the proposed method

The method proposed in section 4 is now applied on the digital system. Because no structure has to be cleared, it starts with the definition of set  $F_1$ , which is composed with the following formulae:

$$f_1 \text{ with } s(f_1) = \sqcup \{x_0, x_1\} \quad f_6 \text{ with } s(f_6) = \sqcup \{x_1\}$$

$$\begin{array}{ll} f_2 \text{ with } s(f_2) = \sqcup \{x_1, x_2\} \quad f_7 \text{ with } s(f_7) = \sqcup \{x_2\} \\ f_3 \text{ with } s(f_3) = \sqcup \{x_2, x_3\} \quad f_8 \text{ with } s(f_8) = \sqcup \{x_3\} \\ f_4 \text{ with } s(f_4) = \sqcup \{x_3, x_4\} \quad f_9 \text{ with } s(f_9) = \sqcup \{x_4\} \\ f_5 \text{ with } s(f_5) = \sqcup \{x_0\} \end{array}$$

The variable  $x_0$  is one of the variables with the lowest order in  $F_1$ . It is selected first. The operator  $\bowtie_{x_0}$  is then used for all formulae where it applies. Then, removing the formulae containing the variable  $x_0$ , the following set of formulae  $F$  is obtained:

$$\begin{array}{l} f_1 \bowtie_{x_0} f_5 \text{ with } s(f_1 \bowtie_{x_0} f_5) = \sqcup \{x_1\} \\ f_2 \text{ with } s(f_2) = \sqcup \{x_1, x_2\} \\ f_3 \text{ with } s(f_3) = \sqcup \{x_2, x_3\} \\ f_4 \text{ with } s(f_4) = \sqcup \{x_3, x_4\} \\ f_6 \text{ with } s(f_6) = \sqcup \{x_1\} \\ f_7 \text{ with } s(f_7) = \sqcup \{x_2\} \\ f_8 \text{ with } s(f_8) = \sqcup \{x_3\} \\ f_9 \text{ with } s(f_9) = \sqcup \{x_4\} \end{array}$$

Then,  $x_4$  is selected. The operator  $\bowtie_{x_4}$  is then used for all formulae where it applies. Then, removing the formulae containing the variable  $x_4$ , the following set of formulae  $F$  is obtained:

$$\begin{array}{l} f_1 \bowtie_{x_0} f_5 \text{ with } s(f_1 \bowtie_{x_0} f_5) = \sqcup \{x_1\} \\ f_2 \text{ with } s(f_2) = \sqcup \{x_1, x_2\} \\ f_3 \text{ with } s(f_3) = \sqcup \{x_2, x_3\} \\ f_4 \bowtie_{x_4} f_9 \text{ with } s(f_4 \bowtie_{x_4} f_9) = \sqcup \{x_3\} \\ f_6 \text{ with } s(f_6) = \sqcup \{x_1\} \\ f_7 \text{ with } s(f_7) = \sqcup \{x_2\} \\ f_8 \text{ with } s(f_8) = \sqcup \{x_3\} \end{array}$$

Then,  $x_1, x_2$  or  $x_3$  can be selected because their order is 3. Let us choose  $x_1$ . The operator  $\bowtie_{x_1}$  is then used for all formulae where it applies. Then, removing the formulae containing the variable  $x_1$ , the following set of formulae  $F$  is obtained:

$$\begin{array}{l} f_3 \text{ with } s(f_3) = \sqcup \{x_2, x_3\} \\ f_4 \bowtie_{x_4} f_9 \text{ with } s(f_4 \bowtie_{x_4} f_9) = \sqcup \{x_3\} \\ f_7 \text{ with } s(f_7) = \sqcup \{x_2\} \\ f_8 \text{ with } s(f_8) = \sqcup \{x_3\} \\ (f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2 \text{ with } s((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) = \sqcup \{x_2\} \\ (f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6 \text{ with } s((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6) = \sqcup \emptyset \\ f_2 \bowtie_{x_1} f_6 \text{ with } s(f_2 \bowtie_{x_1} f_6) = \sqcup \{x_2\} \end{array}$$

Formula  $(f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6$  is an MPTF. Then, variable  $x_3$  should be selected. The operator  $\bowtie_{x_3}$  is then used for all formulae where it applies. Then, removing the formulae containing the variable  $x_3$ , the following set of formulae  $F$  is obtained:

$$\begin{array}{l} (f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6 \text{ with } s((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6) = \sqcup \emptyset \\ f_7 \text{ with } s(f_7) = \sqcup \{x_2\} \\ (f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2 \text{ with } s((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) = \sqcup \{x_2\} \\ f_2 \bowtie_{x_1} f_6 \text{ with } s(f_2 \bowtie_{x_1} f_6) = \sqcup \{x_2\} \\ f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9) \text{ with } s(f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9)) = \sqcup \{x_2\} \\ f_3 \bowtie_{x_3} f_8 \text{ with } s(f_3 \bowtie_{x_3} f_8) = \sqcup \{x_2\} \\ (f_4 \bowtie_{x_4} f_9) \bowtie_{x_3} f_8 \text{ with } s((f_4 \bowtie_{x_4} f_9) \bowtie_{x_3} f_8) = \sqcup \emptyset \end{array}$$

A new MTPF has been found:  $(f_4 \bowtie_{x_4} f_9) \bowtie_{x_3} f_8$ . Only variable  $x_2$  remains. The operator  $\bowtie_{x_2}$  is then used for all formulae where it applies. Then, removing the formulae containing the variable  $x_2$ , the resulting structures are:

- $(f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6 \text{ with } s((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6) = \sqcup \emptyset$

- $(f_4 \bowtie_{x_4} f_9) \bowtie_{x_3} f_8$  with  $s((f_4 \bowtie_{x_4} f_9) \bowtie_{x_3} f_8) = \perp \emptyset \perp$
- $f_7 \bowtie_{x_2} ((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2)$  with  $s(f_7 \bowtie_{x_2} ((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2)) = \perp \emptyset \perp$
- $f_7 \bowtie_{x_2} (f_2 \bowtie_{x_1} f_6)$  with  $s(f_7 \bowtie_{x_2} (f_2 \bowtie_{x_1} f_6)) = \perp \emptyset \perp$
- $f_7 \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))$  with  $s(f_7 \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))) = \perp \emptyset \perp$
- $f_7 \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)$  with  $s(f_7 \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)) = \perp \emptyset \perp$
- $((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))$  with  $s(((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))) = \perp \emptyset \perp$
- $((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)$  with  $s(((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)) = \perp \emptyset \perp$
- $(f_2 \bowtie_{x_1} f_6) \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))$  with  $s((f_2 \bowtie_{x_1} f_6) \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))) = \perp \emptyset \perp$
- $(f_2 \bowtie_{x_1} f_6) \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)$  with  $s((f_2 \bowtie_{x_1} f_6) \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)) = \perp \emptyset \perp$

Formulae  $((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) \bowtie_{x_2} (f_2 \bowtie_{x_1} f_6)$  and  $(f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9)) \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)$  have been calculated and then removed because each the formulae  $(f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6$  and  $(f_4 \bowtie_{x_4} f_9) \bowtie_{x_3} f_8$  are respectively simpler. Only 17 elementary propagations have been necessary to find all the MPTF instead of 119 with Krysander's approach and 248 ways of propagations with the bipartite graph approach.

## 6. CONCLUSION

This paper formalizes structural modeling using tools inspired from relational algebra and shows how it can support engineers in computing detection tests for the system to be diagnosed. Relational algebra abstracts propagations and makes it possible to find testable subsystems using a less intuitive but more efficient approach. In this paper, a heuristic consisting in removing variables with lowest order first is proposed. The proposed method provides the constraints to be used to design each test, but also a way of combining one each other.

The proposed procedure has been designed in order to reduce as much as possible the number of computations. It has been compared to two alternative approaches and significant improvements have been pointed out: it requires less computations and handles deductibility.

Thanks to relational algebra and structural modeling, it has been shown that very specific contexts, such as non deductible variables, can be handled. Comparing to existing approaches, the proposed one is very flexible.

## REFERENCES

- M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-tolerant Control*. Springer-Verlag, 2006.
- J.P. Cassar and M. Staroswiecki. A structural approach for the design of failure detection and identification

- systems. In *IFAC, IFIP,IMACS Conference on control of industrial Systems*, pages 329–334, Belfort, France, 1997.
- L. Chittaro and R. Ranon. Hierarchical model-based diagnosis based on structural abstraction. *Artificial Intelligence*, 1-2:147–182, 2004.
- E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13:377–387, 1970.
- P. Dague. Théorie logique du diagnostic à base de modèles. In B. Dubuisson, editor, *Diagnostic, Intelligence artificielle et reconnaissance de formes*, pages 17–104. Hermès Science, Paris, 2001.
- R. Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410, 1984.
- J. De Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.
- P. Declerck and M. Staroswiecki. Characterization of the canonical components of a structural graph for fault detection in large scale industrial plants. In *European Control Conference*, pages 298–303, Grenoble, France, 1991.
- A. L. Dulmage and N. S. Mendelsohn. A structure theory of bi-partite graphs of finite exterior extension. *Transactions of the Royal Society of Canada*, 53(III): 1–13, 1959.
- D. Dustegor, E. Frisk, V. Cocquempot, M. Krysander, and M. Staroswiecki. Structural analysis of fault isolability in the DAMADICS benchmark. *Control Engineering Practice*, 14:597–608, 2006.
- P. M. Frank. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - a survey and some new results. *Automatica*, 26(3):459–471, 1990.
- E. Frisk. Residual generator for non-linear polynomial systems - a grobner basis approach. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, 2000.
- M. Krysander, J. Aslund, and M. Nyberg. An efficient algorithm for finding over-constrained sub-systems for construction of diagnostic tests. In *16th International Workshop on Principles of Diagnosis (DX-05)*, 2005.
- M. Krysander, J. Åslund, and M. . Nyberg. An efficient algorithm for finding minimal overconstrained subsystems for model-based-diagnosis. *IEEE Transactions on systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(1):197–206, 2008.
- P. Mishra and M.H. Eich. Join processing in relational databases. *ACM Computing Surveys*, 24(1):63–113, 1992.
- P. P. Nayak and A. Y. Levy. A semantic theory of abstractions. In *14th International Joint Conference on Artificial Intelligence IJCAI-95*, pages 196–203, Montreal, Canada, 1995.
- M. Nyberg and M. Krysander. Combining ai, fdi, and statistical hypothesis-testing in a framework for diagnosis. In *IFAC Safeprocess'03*, Washington, U.S.A., 2003.
- R. Patton, P. Frank, and R. Clark (Eds). *Fault diagnosis in dynamic systems*. International series in systems and control engineering. Prentice Hall, 1989.
- S. Ploix, S. Touaf, and J. M. Flaus. A logical framework for isolation in fault diagnosis. In *SAFEPROCESS'2003*, Washington D.C., U.S.A., 2003.

- S. Ploix, M. Desinde, and S. Touaf. Automatic design of detection tests in complex dynamic systems. In *16th IFAC World Congress*, Prague, Czech republic, 2005. 4-8 july.
- S. Ploix, A. Yassine, and J.-M. Flaus. An improved algorithm for the design of testable subsystems. In *The 17th IFAC World Congress*, Seoul, Corea, 2008.
- B. Pulido and C. Alonso. Possible conflicts, arrs, and conflicts. In *13th International Workshop on Principles of Diagnosis (DX02)*, pages 122–128, May 2002.
- R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
- L. Travé-Massuyès, T. Escobet, and X. Olive. Diagnosability analysis based on component supported analytical redundancy relations. *IEEE Transactions*, 36:1146 – 1160, 2006a.
- L. Travé-Massuyès, T. Escobet, and X. Olive. Diagnosability analysis based on component supported analytical redundancy relations. *IEEE transactions on Systems, Man, And Cybernetics - Part A: Systems and Humans*, 36(6):1146–1160, November 2006b.
- A. Willsky. A survey of design methods for failure detection in dynamic systems. *Automatica*, 21:601–611, 1976.
- A. Yassine, S. Ploix, and J.-M. Flaus. A method for sensor placements taking into account diagnosability criteria. *Applied Mathematics and Computer Science*, 18(4), 2008.