Int. J. Appl. Math. Comput. Sci., 2010, Vol. 20, No. 1, 175–190
DOI: 10.2478/v10006-010-0013-7

A NEW EFFICIENT AND FLEXIBLE ALGORITHM FOR THE DESIGN OF TESTABLE SUBSYSTEMS

STÉPHANE PLOIX, ABED ALRAHIM YASSINE, JEAN-MARIE FLAUS

G-SCOP lab, 46 avenue Félix Viallet, 38 031 Grenoble, France
e-mail: {stephane.ploix,jean-marie.flaus}@inpg.fr,
abed-alrahim.yassine@g-scop.inpg.fr

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 that can lead to various types of detection tests. This method, which is based on structural analysis, provides the constraints that have to be used for the design of each detection test and manages 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. Application examples dealing with a road network, a digital counter and an electronic circuit are presented.

Keywords: automatic test design, structural approach, fault diagnosis, analytical redundancy relations, relational algebra.

1. Introduction

Generally speaking, diagnostic analysis of physical systems relies on detection tests that make it possible to detect abnormal behavior. In the scientific literature, there are two main trends for diagnostic analysis. The first one comes from the artificial intelligence community (Reiter, 1987; De Kleer and Williams, 1987; Ligęza and Górny, 2000; Górny and Ligęza, 2001; 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 observations. This approach raises a practical problem: in the industrial world, even if the constraints can be modeled, the tests usually do not result from simple consistency tests between constraints and data, especially in dynamic systems modeled by differential equations. Usually, constraints model only a part of the knowledge: noise and modeling uncertainties are not taken into account in the constraints *a priori* but only *a posteriori.* Consequently, an engineer usually only needs to know the sets of constraints that can lead to tests before designing and validating test algorithms.

The second stream coming from the fault detection and isolation community (Willsky, 1976; Patton *et al*., 1989; Frank, 1990; Cassar and Staroswiecki, 1997) 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 in 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 founded on state space representations or on Grobner bases (Frisk, 2000). In diagnosis, structural modeling was introduced in (Davis, 1984). Using a semantic theory of abstraction (Nayak and Levy, 1995; Chittaro and Ranon, 2004), it was 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*., 2003; Cassar and Staroswiecki,

1997; Declerck and Staroswiecki, 1991). Finding testable sets of constraints may indeed be achieved thanks to an elimination procedure that combines constraints in order to eliminate all the unknown physical variables and therefore get constraints containing only known data, i.e., testable constraints. However, as shown in this paper, this approach leads to a high level of complexity when searching all the testable sets of constraints. In (Travé-Massuyès *et al*., 2006), an alternative method was proposed, but the level of complexity is still high. Krysander *et al.* (2005) proposed an improved algorithm which, however, does not manage constraints containing non-deductible variables.

Propagation algorithms, such as value propagation (Fron, 1994) or constraint propagation (Russell and Norvig, 2003; Dechetr, 2003), may also be considered as candidate solutions to compute testable subsystems. If it is possible to propagate, it is possible to pre-compute propagation paths. Pre-computation of a propagation path corresponds to a testable set of constraints, which leads to a test, sometimes called an Analytical Redundancy Relation (ARR). However, in diagnosis, there is usually a high level of redundancy in information. Therefore, many propagations have to be done: some may be consistent with data, others may not, but each propagation provides a symptom that is meaningful for diagnosis. Performing all the propagations seems like a solution. However, some propagations are equivalent or include others and it is not possible to detect this during the propagation phase. Pre-computation of propagation paths is required to avoid redundant computations.

This paper presents a general method that provides testable subsystems. A new 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 proposed. It relies also on a structural abstraction of the constraints and traces all the constraints that are involved in testable subsystems, thus making it possible to determine what physical components are examined by each test. A computational tool has been developed and examples of applications are presented in order to show the performance of the algorithm.

2. Relational algebra and join-operator

Relational Algebra (RA) can be viewed as a data manipulation language for a relational model. It consists of several basic operations which make it possible to specify retrieval requests. These operations are applied to several relations (constraints) in order to produce new relationships.

Three sets of relational operators are usually distinguished:

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

A relation R may be a class or a table. It is denoted by $R(A_1, A_2, \ldots, A_n)$ such that $A_1, A_2, \ldots, A_n$ represent the attributes of the relationship R. A value domain $val(A_i)$ corresponds to each attribute A_i. Consequently, the relation $R(A_1, A_2, \ldots, val(A_n))$ corresponds to a set of tuples $t_i = (a_{1,i}, a_{2,i}, \ldots, a_{n,i}) \subseteq val(A_1) \times val(A_2) \times \cdots \times A_n$. Further on, 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 $R(A_1, A_2, \ldots, A_n)$ that satisfy a logical condition defined over $A_1 \times A_2 \times \cdots \times A_n$. It is denoted $\nabla_E R$.

Definition 2. (Projection)
A *projection* of a relation $R(A_1, A_2, \ldots, A_n)$ according to some attributes $\{A^1, A^2, \ldots, A^m\} \subseteq \{A_1, A_2, \ldots, A_n\}$ amounts to keeping the tuples corresponding to attributes $A_{m+1}, \ldots, A_n$ and removing the resulting duplicated tuples.

Definition 3. (Cartesian product)
The *Cartesian* product of two relations 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 by $R_1 \times R_2 = \{(r_i, r_j) : r_i \in R_1 \text{ and } r_j \in R_2\}$

Definition 4. (Join operator)
The *join operator* is an 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.

The general join is called a theta-join. The theta operator can be one of the following: $=, \neq, \leq, \geq, <, >$. The attributes used to define the join condition must be comparable using the theta operator. In its most general form, the join condition E consists of multiple simple conditions of the form described above. The join operation may be said to be equivalent to a Cartesian product followed be a select: $R_1 \bowtie_E R_2 = \nabla_E(R_1 \times R_2)$.

Definition 5. (Equi-join)
An *equi-join* is a theta-join such that the condition join E is an equality between an attribute A_1 of the relationship R_1 and an attribute A_2 of the relationship R_2. The equi-join is denoted by $R_1 \bowtie_{A_1=A_2} R_2$.

Indeed, the equi-join is a Cartesian product followed by a select: $R_1 \bowtie_{A_1=A_2} R_2 = \nabla_{A_1=A_2}(R_1 \times R_2)$.

Consider, for example, two relationships R_1 et R_2 presented in Fig. 1. The Cartesian product and the equi-join are given by the relationships R_3 and R_4.

Relation : R_1

A	B
1	7
4	2
7	3

Relation : R_2

C	D	E
3	2	5
7	1	6
5	6	4
2	8	1

carestian product : R_3

A	B	C	D	E
1	7	3	2	5
1	7	7	1	6
1	7	5	6	4
1	7	2	8	1
4	2	3	2	5
4	2	7	1	6
4	2	5	6	4
4	2	2	8	1
7	3	3	2	5
7	3	7	1	6
7	3	5	6	4
7	3	2	8	1

equijoin : R_4

A	B	C	D	E
1	7	7	1	6
4	2	2	8	1
7	3	3	2	5

Fig. 1. Example of a Cartesian product.

The concept of value propagation between two constraints k_i and k_j (such that a constraint k is a relation which models the behavior of a component c) is similar to the projection of the equi-join of the two constraints. The propagation of a value v between the two constraints k_i and k_j may be represented by the join operator: $s_i \bowtie_v s_j = \Pi_{(var(s_i) \cup var(s_j) \setminus \{v\})}(s_i \bowtie_{s_i.v=s_j.v} s_j)$ such that s_i and s_j represent the structures of the constraints k_i and k_j, $var(s_i)$ and $var(s_j)$ represent the sets of variables of the two structures s_i and s_j.

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

3. Problem statement

This section introduces the concepts and the formalism used in the paper. In order to manage testable subsystem design, let us introduce a formalism for behavioral modeling before considering structural modeling.

3.1. Behavioral modeling. Behavioral knowledge starts with physical variables. A physical variable is a potentially observable element of information about the actual state of a system. It is modeled by an implicitly time-varying variable, which has to be distinguished from a parameter that is model-dependent. Generally speaking, even if a physical variable is observable, it is not possible to merge it with data because in fault diagnosis data are only known provided that some actuators or sensors behave properly. Physical variables $V(t) = \{\dots, v_i(t), \dots\}$ are modeled by a space $\mathcal{F}(T, V) = \{V(t); t \in T\}$, where T stands for continuous or discrete sets of time. At any given time t in T, these physical variables belong to a domain $dom(t, V) = dom(V(t))$ representing all the possible values that may have physical variables. Consequently, when considering all $t \in T$, $\{dom(V(t)); t \in T\}$ represents a tube in the physical variable space $\mathcal{F}(T, V)$.

A *data flow* models data provided by a source of information concerning a physical variable. These data can come from measurements or from set points related to controlled variables. A data flow concerning a physical variable v is denoted by $obs(t, v)$ with $obs(t, v) \in dom(v(t))$. It corresponds to a trajectory belonging to the tube $\{dom(v(t)); t \in T\}$ (see Fig. 2). When information about v is coming from different sources, the different data flows can be denoted by $obs_i(t, v)$. Formally, a data flow provided by a component c can be linked to a physical variable: $ok(c) \rightarrow \forall t \in T, obs(t, v) = v$, which means that, if the component named c is in the state ok, then the data $obs(t, v)$ correspond to the actual value of the physical variable v at any time $t \in T$.

For testable subsystem design, it should be noted that all the physical variables have to be considered unknown. Therefore, a mapping is testable if it does not contain any variable standing for physical variables but only data flows. Then, after having identified physical variables, cause-effect relationships between them have been made up. Each relationship between physical variables corresponds to a subset of a domain that models one or several behavioral modes (de Kleer and Williams, 1992; Struss, 1992). In a diagnostic problem, the behavioral modes of a component $modes(c)$ must be a complete set of modes, i.e., at any given time, the state of a component c is assumed to correspond to one and only one mode from $modes(c)$. Most common sets are $modes(c) = \{ok(c), \neg ok(c)\}$, but more complex modes may also be considered: $modes(c) = \{ok(c), fault_1(c), \dots, fault_n(c), otherFault(c)\}$.

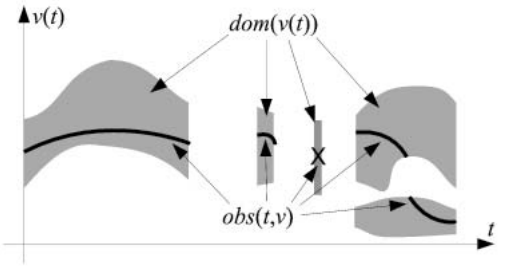

Fig. 2. Physical variable space, tube and trajectory.

Each mode can be linked to a subset $\mathcal{K} \subset \mathcal{F}(T, V)$, which is often referred to as a constraint. Below,

the following notation is adopted: $var(\mathcal{K}) = V$ and $dom(t, \mathcal{K}) = dom(t, var(\mathcal{K}))$. Then, generally speaking, denoting by $\tilde{V}$ a trajectory in the domain $dom(t, \mathcal{K})$, behavioral modeling can be formalized, given the data flows, by

$$mode_1(c_1) \wedge mode_2(c_2) \wedge \cdots \rightarrow \exists \tilde{V} \in \mathcal{K}, \quad (1)$$

where c_1, c_2, ... constitute components and $mode_1 \in modes(c_1)$, $mode_2 \in modes(c_2)$, ...

In order to represent mathematical constraints, mappings are used. For instance, consider the constraint $u - R \times i = 0$. It leads to two obvious following mappings: $u = R \times i$ and $i = (1/R) \times u$. Indeed, designing testable subsystems *a priori* requires combining mappings: it is not possible to check if values may satisfy a set of constraints because data flows are not available during the design stage. Because, at any time, they lead only to one value, mapping ensures that combinations may be made without loss. Note that, for the sake of simplicity, only time invariant mappings are considered here. Generally speaking, a mapping over $dom(t, V)$ is defined from one subspace $dom(t, V_1)$ to another $dom(t, V_2)$, where $\{V_1, V_2\}$ is a partition of V. Note that several mappings κ_i may model the same constraint $\mathcal{K}$. If $\kappa_i : dom(t, V_1) \mapsto dom(t, V_2)$ is a mapping modeling $\mathcal{K}$, the description (1), given the data flows, becomes

$$mode_1(c_1) \wedge mode_2(c_2) \rightarrow \exists \tilde{V_1} \in dom(t, V_1), \exists \tilde{V_2} \in dom(t, V_2)/\tilde{V_2} = \kappa_i(\tilde{V_1}). \quad (2)$$

Therefore structural abstraction must be considered.

3.2. Structural modeling. Generating testable subsystems means that the descriptions (2) are combined, i.e., left-hand-side modes are combined with conjunctions and variables that stand for physical variables are removed when combining mappings so that only data flows are kept. A general method must abstract mappings away as they may be of very different types, ranging from automate to differential equations or qualitative modeling.

This is why structural modeling has to be considered although this advantage is offset by the fact that results may overestimate the effective testable subsystems because knowledge about mappings is not fully taken into account. Overestimation leads to testable subsystems that refer to many more combined constraints than necessary. A consequence is that, during test implementation, verification is needed to see whether all the constraints involved in a testable subsystem are effectively required.

As an abstraction of a mapping, the notion of *structure of constraint* is introduced: basically, the *structure of a constraint* $\mathcal{K}$ corresponds to the variables $var(\mathcal{K})$. Nevertheless, a given constraint may be modeled by different mappings.

Firstly, although mappings to multidimensional spaces could be used, they are difficult to manage in testable subsystem design. It is better to break them down into equivalent sets of one-dimensional mappings. In the following, one-dimensional mappings modeling a constraint $\mathcal{K}$ are named a *realization* of $\mathcal{K}$.

Moreover, several realizations of a constraint may be equivalent. Let κ_i be a realization from $V \backslash \{v\}$ to $\{v\}$. There may be equivalent realizations defined on V that also model the constraint. Therefore, the notion of the *structure of a constraint* can be extended to represent all the equivalent realizations representing a given constraint $\mathcal{K}$. For instance, consider a realization κ modeling a *logical xor*: $x_3 = x_1 \otimes x_2$. There is a mapping leading to x_3 from x_1 and x_2 but also a mapping from x_2 and x_3 leading to x_1 and another one to x_2. In that case, these variables can be qualified as deductible or, when it is meaningful, as calculable with reference to (Iwasaki and Simo, 1994). This is denoted by $var(\kappa) = var^+(\kappa) = \{x_1, x_2, x_3\}$. It is different for a realization κ' modeling a *logical or*: $x_3 = x_1 \vee x_2$ where only x_3 is deductible. This is denoted by $var(\kappa) = var^+(\kappa') \cup var^-(\kappa')$ with $var^+(\kappa) = \{x_3\}$ and $var^-(\kappa) = \{x_1, x_2\}$. The set of equivalent realizations modeling a constraint $\mathcal{K}$ is denoted by $K(\mathcal{K})$. Therefore, the structure of a constraint models at the same time the variables of its physical variable space and a set of possible equivalent 1-dimensional mappings that model this constraint. A structure may represent either a particular realization or a set of equivalent realizations.

The variables $V = var(\mathcal{K})$ can be broken down into a set of deductible variables and a set of non-deductible variables. A structure s will be written as $\llcorner V^-, V^+ \lrcorner$, where V^- and V^+ satisfy $V^- \cap V^+ = \emptyset$. Therefore, the structure modeling a constraint $\mathcal{K}$ is denoted by $s(\mathcal{K}) = \llcorner V^-, V^+ \lrcorner$. Because $\forall v \in V^+$, there is a realization from $K(\mathcal{K})$ leading to v:

$$\forall v \in V^+, \exists \llcorner (V^- \cup V^+) \backslash \{v\}, \{v\} \lrcorner.$$

For the sake of simplicity, the following notation is adopted: $\llcorner \emptyset, V^+ \lrcorner = \llcorner V^+ \lrcorner$ 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 by $s = \llcorner \lrcorner$.

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

Definition 6. A *terminal structure* s satisfies $card(var(s)) = 1$. 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 7. A structure of a constraint 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 by $s_1 \supseteq s_2$.

In order to show that values can be propagated between mappings, i.e., that the intersection of two constraints can be projected without any loss, a join operator is defined. It formalizes the elimination of a variable. Thanks to this formalization, it is possible to abstract the process of elimination and to find eliminations that cannot be easy to find using graph theory. As a prerequisite, the notion of a propagable variable has to be introduced.

Definition 8. Let s_1 and s_2 be two structures related to $\kappa_1 \in K(\mathcal{K}_1)$ and $\kappa_2 \in K(\mathcal{K}_2)$. The propagation of a variable v between $s_1 = s_1(\kappa_1)$ and $s_2 = s_2(\kappa_2)$ is possible only if $v \in var(s_1) \cap var(s_2)$ and if v is deductible in at least one structure. If this condition is satisfied, v is qualified as propagable between s_1 and s_2. By extension, v is also said to be *propagable* between κ_1 and κ_2 and also between $K(\mathcal{K}_1)$ and $K(\mathcal{K}_2)$.

The join operator can now be defined.

Definition 9. Let s_1 and s_2 be two structures, with $V_1^+ = var^+(s_1)$, $V_1^- = var^-(s_1)$, $V_2^+ = var^+(s_2)$ and $V_2^- = var^-(s_2)$. The join operator, denoted by $\bowtie_v$, where v is a propagable variable between s_1 and s_2, is defined only in the following two situations:

- if $\{v\} \in V_1^+ \cap V_2^-$, then
  $s_1 \bowtie_v s_2$
  $= \llcorner \left(V_1^- \cup V_1^+ \cup V_2^-\right) \setminus \left(V_2^+ \cup \{v\}\right), V_2^+ \lrcorner$,
- if $\{v\} \in V_1^+ \cap V_2^+$, then
  $s_1 \bowtie_v s_2$
  $= \llcorner \left(V_1^- \cup V_2^-\right) \setminus \left(V_1^+ \cup V_2^+\right), \left(V_1^+ \cup V_2^+\right) \setminus \{v\} \lrcorner$.

If a formula $s_1 \bowtie_v s_2$ satisfies one of the previous two 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 realization, 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 sets of equivalent realizations. Then a wrapping structure of the set of equivalent realizations resulting from the propagation of a variable v between K_1 and K_2 can be obtained using the $\bowtie_v$ operator on $s(K_1)$ and $s(K_2)$.*

Proof. Write

$$s(K_1) = \llcorner V_1^-, V_1^+ \lrcorner,\ s(K_2) = \llcorner V_2^-, V_2^+ \lrcorner$$

and let K be the set of equivalent realizations resulting from the propagation of v between K_1 and K_2. The set K depends on the presence of other propagable variables. If v is the only propagable variable between K_1 and K_2, the exact structure of K can be deduced; otherwise, only a wrapping structure can be found.

Consider the situation where there is only one propagable variable $v \in V_1^+ \cap V_2^-$. Then there is a $\kappa_{1,v} \in K_1$ such that $v = \kappa_{1,v}\left(\left(V_1^+ \backslash \{v\}\right) \cup V_1^-\right)$ and v can be propagated into K_2. If $V_2^+ \neq \emptyset$, $\forall w \in V_2^+$, $\exists \kappa_{2,w} \in K_2 / w = \kappa_{2,w}\left(\left(V_2^+ \backslash \{w\}\right) \cup V_2^-\right)$. Because $v \in V_2^-$, it yields that there is a κ_w such that

$$\begin{aligned} w = &\kappa_w\big(\left(\left(V_2^+ \backslash \{w\}\right) \cup \left(V_2^- \backslash \{v\}\right)\right) \\ &\cup \left(\left(V_1^+ \backslash \{v\}\right) \cup V_1^-\right)\big). \end{aligned}$$

Because v is the only propagable variable, $w \notin \left(V_1^+ \cup V_1^-\right)$ and therefore, κ_w is a realization. Because this result is true, $\forall w \in V_2^+$, the structure of the resulting set of equivalent realizations is $\llcorner (V_1^- \cup V_1^+ \cup V_2^-) \backslash (V_2^+ \cup \{v\}), V_2^+ \lrcorner$. Moreover, if V_2^+ is empty, the previous result remains true: $\llcorner (V_1^- \cup V_1^+ \cup V_2^-) \backslash \{v\}, \emptyset \lrcorner$. It yields $\forall v \in V_1^+ \cap V_2^-$, $s(K) = s(K_1) \bowtie_v s(K_2)$.

If the unique propagable variable satisfies $v \in V_1^+ \cup V_2^+$, additional deductible variables can be found. Indeed,

- if $V_2^+ \backslash \{v\} \neq \emptyset$, $\exists \kappa_{1,v} \in K_1$ such that
  $v = \kappa_{1,v}\left(\left(V_1^+ \backslash \{v\}\right) \cup V_1^-\right)$
  and v can be propagated into K_2:
  $\forall w \in V_2^+ \backslash \{v\}, \exists \kappa_{2,w} \in K_2 / w$
  $= \kappa_{2,w}\left(\left(V_2^+ \backslash \{w\}\right) \cup V_2^-\right)$.
  Because $v \in V_2^+$, it yields
  $\exists \kappa_w / w = \kappa_w\big(\left(\left(V_2^+ \cup V_1^+\right) \backslash \{v, w\}\right)$
  $\cup V_1^- \cup V_2^-\big)$;
- if $V_1^+ \backslash \{v\} \neq \emptyset$, $\exists \kappa_{2,v} \in K_2$ such that
  $v = \kappa_{2,v}\left(\left(V_2^+ \backslash \{v\}\right) \cup V_2^-\right)$
  and v can be propagated into K_1:
  $\forall w \in V_1^+ \backslash \{v\}, \exists \kappa_{1,w} \in K_1 / w$
  $= \kappa_{1,w}\left(\left(V_1^+ \backslash \{w\}\right) \cup V_1^-\right)$.
  Because $v \in V_1^+$, it yields
  $\exists \kappa_w / w = \kappa_w\big(\left(\left(V_1^+ \cup V_2^+\right) \backslash \{v, w\}\right)$
  $\cup V_1^- \cup V_2^-\big)$.

Because v is the only propagable variable, it yields that κ_w is a realization and then that the structure of the resulting constraint is given by $\llcorner (V_1^- \cup V_2^-) \backslash (V_1^+ \cup V_2^+), (V_1^+ \cup V_2^+) \backslash \{v\} \lrcorner$. Consequently, $\forall v \in V_1^+ \cap V_2^+$, $s(K) = s(K_1) \bowtie_v s(K_2)$. These results remain true if V_1^+ or V_2^+ are empty.

When there are several propagable variables, it is no longer possible to prove that the functions κ_w are realizations and hence the previous results become

- $\forall v \in V_1^+ \cap V_2^-$, $s(K) \subseteq s(K_1) \bowtie_v s(K_2)$,
- $\forall v \in V_1^+ \cap V_2^+$, $s(K) \subseteq s(K_1) \bowtie_v s(K_2)$.

■

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 mappings. The way to use these tools is described in the next section.

4. Testable subsystem design

A testable subsystem is a set of constraints which leads to an analytical redundancy relation. An ARR is a static or dynamic constraint which links time evolution of known variables when the system operates according to its normal operation model.

Some basic concepts have to be first introduced before tackling the design of testable subsystems. Then, some particular modeling contexts are examined.

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 subformulas linked to the join operator, where the elementary formulas are structures. Thanks to this operator, a propagation formula f can be evaluated as a structure $s(f)$. The set of all the formulas is denoted by $\mathbb{F}$.

Taking into account that the most basic formula looks like $s_i \bowtie_v s_j$, the definition of an evaluable formula can be introduced.

Definition 10. A formula is qualified as *evaluable* if all its subformulas and the formula itself are evaluable.

Definition 11. The *support* of a formula $f \in \mathcal{F}$, denoted by $\sigma(f)$, is the set of all the structures involved in the formula.

Definition 12. The *degree* of a formula $f \in \mathbb{F}$, denoted by $d(f)$, is equal to the number of times the join operator appears in the formula—it represents the number of elementary propagations.

Definition 13. Two formulas f_1 and f_2 are comparable if $\sigma(f_1) = \sigma(f_2)$ and if $var(s(f_1)) = var(s(f_2))$, i.e., they have the same constraints and the same variables. This is denoted by $f_1 \sim f_2$.

Moreover, during propagations, a given variable can be instantiated only once. However, in some formulas, 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 14. A formula f_1 is *simpler* than a formula f_2 if

- $\sigma(f_1) \subset \sigma(f_2)$ and $var(s(f_1)) \subset var(s(f_2))$, or
- $f_1 \sim f_2$ and $d(f_1) < d(f_2)$, or
- $f_1 \sim f_2$ and $d(f_1) = d(f_2)$ and $var^+(s(f_1)) \supset var^+(s(f_2))$.

This is denoted by $f_1 \prec f_2$. 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 formulas correspond to wrapping structures of constraints that can be found by combining elementary constraints $\mathbb{K} = \{\dots, \mathcal{K}_i, \dots\}$ of a diagnostic problem. Therefore, the formulas 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 formulas 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 formulas is therefore reduced by one. Applying the join operator $\bowtie_v$ on a variable v belonging to three formulas f_i, f_j and f_k yields three new formulas: $f_i \bowtie_v f_j$, $f_i \bowtie_v f_k$ and $f_j \bowtie_v f_k$. The number of formulas remains constant. Applying the join operator $\bowtie_v$ on a variable v belonging to four formulas f_i, f_j, f_k and f_l yields six new formulas: $f_i \bowtie_v f_j$, $f_i \bowtie_v f_k$, $f_i \bowtie_v f_l$, $f_j \bowtie_v f_k$, $f_j \bowtie_v f_l$ and $f_k \bowtie_v f_l$.

The number of formulas from a set F where a variable v appears is named the *order* of v in $\sigma(F)$. It is denoted by $o_F(v)$. Figure 3 shows the link between the number of structures 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 the number of formulas low.

4.3. Algorithms for the design of testable subsystems. For the design of Testable SubSystems (TSSs), all the possible MTPFs have to be found because TSSs are given by the support of MTPFs. The principle is to iteratively propagate values until MTPFs are found. But, unlike value propagation, a propagation can be envisaged 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 formulas (Definition 10) are represented by the letter F while the corresponding structures by $s(F) = \{s(f); f \in F\}$. Consider a set of constraints intervening in a diagnostic problem and F_0, the corresponding structures, which are also elementary formulas. Here $s(F_0)$

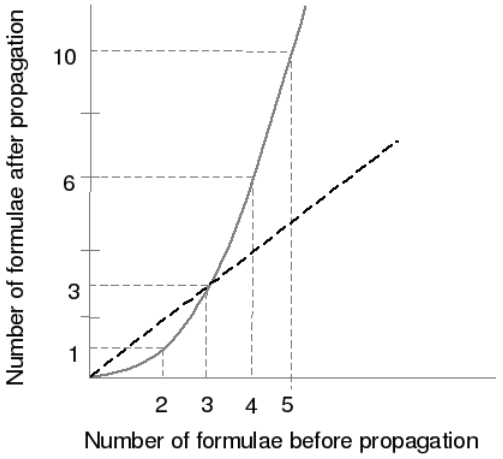

Fig. 3. Influence of the join operator on the number of formulas.

denotes the structures corresponding to F_0. The number of structures from a set $s(F)$ where a variable v appears is named the order of v in $\sigma(F)$. It is denoted by $o_F(v)$

Firstly, a propagation cannot be achieved when a variable appears only once in the structures $s(F_0)$. Therefore, structures containing these variables, and their corresponding formulas, should be removed because they have no usefulness in an MTPF. Nevertheless, when some structures are removed, it is possible to find new variables that appear only once. The procedure is then repeated until no more single-occurrence variables remain. This step is a clearing step. It is summarized by the following algorithm.

Algorithm 1 Remove useless structures

Require: F_0, a set of formulae
  $F \leftarrow F_0$
  repeat
    $V \leftarrow \{v \in var(s(F)); o_F(v) = 1\}$
    $F \leftarrow \{f \in F; var(s(f)) \cap V = \emptyset\}$
  until $V = \emptyset$
  return F

The resulting cleared set is named F_1. The propagations can now be achieved according to the orders of variables. The variables of the lowest order are selected first. Let v be one of these variables. All the formulas where v appears are selected and, using the join operator, new evaluable formulas are then deduced and added to the current set of formulas. Formulas that overestimate others are removed along with formulas that contain v. This procedure is repeated until all the variables have been considered. The remaining structures are then empty and thus all the MTPFs have been found. The procedure is summarized by the algorithm below.

Algorithm 2 Compute MTPF

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

Sometimes, because the number of testable propagation formulas is very high, it is quicker to find only a subset of all the MTPFs. 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 controled can be reduced—propagations may indeed be stopped when a known variable is found. The resulting MTPFs are called basic MTPFs. They are valuable because they constitute a small subset of all the MTPFs that covers all the exploitable structures present in all the MTPFs. Algorithm 2 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.

4.4. Particular modeling contexts. In this section, two particular contexts are examined: dynamic systems and systems with branchings.

4.4.1. Dynamical systems. There are two kinds of dynamic systems depending on the way they are modeled:

- a variable appears several times in a system but at different time stamps, or
- a variable and some of its derivatives or summations (whatever the order is) appear in the system.

The first case mainly concerns time-delays and discrete time recurrent systems. According to Section 3, each variable stands for a tube in a physical variable space. Therefore, a time delay, modeled by $y(t + \Delta) = x(t)$, is a constraint that establishes a link between two tubes: $\{dom(y(t + \Delta)); t \in T\}$ and $\{dom(x(t))\}; t \in T$. Therefore, even if the two variables model the same physical variable in the structural model, they cannot be

merged. Consider now the following discrete-time recurrent model:

$$\begin{aligned} x((k+1)T_e) &= Ax(kT_e) + Bu_k(kT_e), \\ y(kT_e) &= Cx(kT_e), \qquad\qquad k \in \mathbb{N}, \end{aligned}$$

where T_e stands for the sampling period.

The physical variable modeled by x appears twice. Therefore, the constraint must be implicitly completed by a time delay between the variables $x((k+1)T_e)$ and $x(kT_e)$. Structurally speaking, these constraints are modeled by the following structures:

$$\begin{aligned} &\llcorner\{x(kT_e), x((k+1)T_e), u(kT_e)\}\lrcorner, \\ &\llcorner\{x(kT_e), x((k+1)T_e)\}\lrcorner, \\ &\llcorner\{x(kT_e), y(kT_e)\}\lrcorner. \end{aligned}$$

Moreover, if the tube corresponding to $x((k+1)T_e)$ appears only in this constraints (which is usually the case in practice), the join operator $\bowtie_{x((k+1)T_e)}$ can be applied between $\llcorner\{x(kT_e), x((k+1)T_e), u(kT_e)\}\lrcorner$ and $\llcorner\{x(kT_e), x((k+1)T_e)\}\lrcorner$, and it results in

$$\begin{aligned} &\llcorner\{x(kT_e), u(kT_e)\}\lrcorner, \\ &\llcorner\{x(kT_e), y(kT_e)\}\lrcorner. \end{aligned}$$

The second case mainly concerns integrations and differential equations. Consider, for instance, the following model: $\frac{\mathrm{d}x}{\mathrm{d}t} = u$. $\frac{\mathrm{d}x}{\mathrm{d}t}$ corresponds to a tube, which can be connected to x by adding the implicit constraint: $x = \int \frac{\mathrm{d}x}{\mathrm{d}t}\mathrm{d}t$. The initial condition could also be taken into account by considering $x = \int_0^{t_f} \frac{\mathrm{d}x}{\mathrm{d}t}\mathrm{d}t + x_0$. In this case, the structures become $\llcorner\{\frac{\mathrm{d}x}{\mathrm{d}t}, u\}\lrcorner$ and $\llcorner\{x, \frac{\mathrm{d}x}{\mathrm{d}t}, x_0\}\lrcorner$. In the same way as time-delays, the join operator $\bowtie_{\frac{\mathrm{d}x}{\mathrm{d}t}}$ can be applied to obtain the following structure: $\llcorner\{u, x\}\lrcorner$ or, if the initial condition is considered, $\llcorner\{u, x, x_0\}\lrcorner$. This result remains true for summations and derivatives of any order. Let us now consider the ordinary differential equation

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= Ax + Bu, \\ y &= Cx. \end{aligned}$$

Here, too, an implicit constraint has to be added: $x = \int \frac{\mathrm{d}x}{\mathrm{d}t}\mathrm{d}t$. The following structures arise:

$$\begin{aligned} &\llcorner\{x, \frac{\mathrm{d}x}{\mathrm{d}t}, u\}\lrcorner, \\ &\llcorner\{x, \frac{\mathrm{d}x}{\mathrm{d}t})\}\lrcorner, \\ &\llcorner\{x, y\}\lrcorner. \end{aligned}$$

Using the join operator $\bowtie_{\frac{\mathrm{d}x}{\mathrm{d}t}}$, it yields

$$\begin{aligned} &\llcorner\{x, u\}\lrcorner \\ &\llcorner\{x, y\}\lrcorner. \end{aligned}$$

4.4.2. Systems with branchings. Using a structural approach for the design of testable subsystems leads to a general method that may potentially be applied to any kind of systems. The case of dynamic systems have been examined, but another context requires also a special attention—systems with branchings. These systems can be managed by the method presented in Section 4.3, but the amount of computations can be drastically reduced.

Compared with other structural approaches for the generation of a TSS, such cases cannot be taken into account easily. Let us examine how it can be managed with the proposed approach. Consider, for instance, the roads in Fig. 4. Denoting by $R_{v_i v_j}$ a road section linking v_i to v_j, a structural model of the road network is given by

$$\begin{aligned} s_1 &= s(R_{v_1 v_2}) = \llcorner v_1, v_2 \lrcorner, \\ s_2 &= s(R_{v_3 v_5}) = \llcorner v_3, v_5 \lrcorner, \\ s_3 &= s(R_{v_4 v_6}) = \llcorner v_4, v_6 \lrcorner, \\ s_4 &= s(\text{crossroad}) = \llcorner v_2, v_3 \lrcorner, \\ s_5 &= s(\text{crossroad}) = \llcorner v_2, v_4 \lrcorner, \\ s_6 &= s(\text{crossroad}) = \llcorner v_3, v_4 \lrcorner. \end{aligned}$$

Generally speaking, if a car goes from v_1 to v_5, it is unlikely that its route could also pass through v_4. Then, a testable subsystem gathering all the constraints of the crossroad would not make sense. If it is not taken into account, the number of generated TSSs will be too much important. In that simple case, unrealistic routes would appear: $(v_1, v_2, v_3, v_4, v_6)$, $(v_1, v_2, v_4, v_3, v_5)$, $(v_5, v_3, v_2, v_4, v_6)$, ... Imagine now a complete network with many crossroads. Unrealistic combinations of constraints will be recombined and this will provide new unrealistic constraints and so on, until a possibly huge number of unrealistic TSSs is obtained. In order to avoid these physical variables, junctions have to be taken into account during the generation of formulae.

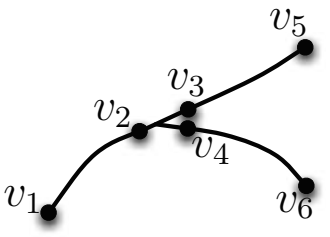

Fig. 4. Simple road network is a system with branchings.

In order to avoid unrealistic TSSs, a set of exclusions can be defined. In order to model that only two constraints of the crossroad can be gathered in one formula, the following exclusion set can be defined: $\{s_4, s_5\}$. This means that these structures cannot appear in the same formula. Generally speaking, a set of exclusion sets, denoted by

$\mathbb{X} = \{\{s_4, s_5\}, \{s_4, s_6\}, \{s_5, s_6\}\}$, can be defined. Thus, during the computations of formulas (Section 4.3), once a new formula is found, it is examined in order to determine if it does not gather all the structures belonging to an exclusion set of $\mathbb{X}$. In that case, the new formula is removed from F'' in Algorithm 2. It then becomes Algorithm 3.

Algorithm 3 Compute MTPF with exclusions

Require: F_1, a set of formulae
Require: $\mathbb{X}$, a set of exclusion sets
  $F \leftarrow F_1$
  while $var(s(F)) \neq \emptyset$ **do**
    select $v \in var(s(F))$ **such that** $o_F(v) \leq o_F(v_i), \forall v_i \in var(s(F))$
    $F' \leftarrow \{f / v \in var(s(f))\}$
    $F'' \leftarrow \{f_i \bowtie_v f_j; (f_i, f_j) \in F'^2, i \neq j, f_i \bowtie_v f_j \, evaluable\}$
    for all $f \in F''$, **do**
      if $\exists x \in \mathbb{X} / x \subset \sigma(f)$ **then**
        $F'' \leftarrow F'' \backslash \{f\}$
      end if
    end for
    $F \leftarrow (F \backslash F') \cup F''$
    $F \leftarrow F \backslash \{f \in F; \exists f_i \in F, f_i \neq f, f \succ f_i\}$
  end while
  return F

Systems with branchings are not rare—they can be encountered in discrete event systems with conveyors, transportation networks, but also in the diagnosis of cognitive skills (Ploix *et al.*, 2004), where different alternatives in reasoning may be adopted.

5. Comparison with other approaches

The bipartite graph-based approach, Travé-Massuyès's approach and Krysanders approach are compared with 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 the existing approaches. Consider a system composed by four iterations of a electronic counter equipped with a digital display. The system is modeled by the following constraints:

$$\begin{aligned}
\mathcal{K}_1 &: \quad x_1 = x_0 + 1, \\
\mathcal{K}_2 &: \quad x_2 = x_1 + 1, \\
\mathcal{K}_3 &: \quad x_3 = x_2 + 1, \\
\mathcal{K}_4 &: \quad x_4 = x_3 + 1, \\
\mathcal{K}_5 &: \quad \tilde{x}_0 = x_0, \\
\mathcal{K}_6 &: \quad \tilde{x}_1 = x_1, \\
\mathcal{K}_7 &: \quad \tilde{x}_2 = x_2, \\
\mathcal{K}_8 &: \quad \tilde{x}_3 = x_3, \\
\mathcal{K}_9 &: \quad \tilde{x}_4 = x_4.
\end{aligned} \tag{3}$$

5.1. Bipartite graph approach. The bipartite graph approach (Dulmage and Mendelsohn, 1959) was used in (Blanke *et al.*, 2003; Declerck and Staroswiecki, 1991; Staroswiecki and Declerck, 1989) to compute analytical redundancy relations—it corresponds to the supports of the MTPF. The bipartite graph approach is a structural one, which only requires the knowledge of the variables that occur in constraints. The bipartite graph is defined by $G = (\mathbb{K}, var(\mathbb{K}), \mathcal{E})$, where $\mathbb{K}$ gathers available sets of equivalent realizations and $\mathcal{E} = \mathbb{K} \times var(\mathbb{K})$ stands for edges in the graph. Each set $K = K(\mathcal{K})$ of equivalent realizations from $\mathbb{K}$ generates the edges $\{(K, v); v \in var(K)\}$. Complete matchings within a connected graph, i.e., a sub-graph containing all the variables $var(\mathbb{K})$ and whose edges contain no common vertex, neither constraint nor variable, are then selected in order to find all the ARRs.

A matching directs the search of ARRs, which are composed of alternated chains starting with a terminal constraint and ending with an unmatched constraint. In a connected bipartite graph, for a given matching, the number of ARRs that can be found is equal to the number of unmatched sets of equivalent realizations. From these constraints, the bipartite graph in Fig. 5 can be drawn up.

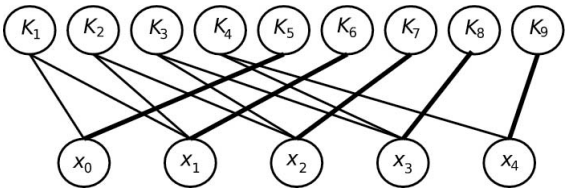

Fig. 5. Bipartite graph with a matching for the counter.

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.*, 2003), four ARRs are obtained. Nevertheless, there are obviously ten ARRs in this example. In order to obtain the other ARRs, 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 a new ARR.

This example shows that a matching makes it possible to obtain only one part of the set of all the ARRs. 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 one, the num-

ber of ARRs that can be found is equal to the number of constraints not matched (four in this example). Therefore, in order to finally keep only ten ARRs, 64×4 redundant ARRs are obtained in this simple example.

The bipartite graph approach induces great complexity even for simple systems. Let us reconsider this example with the join operator.

5.2. Travé-Massuyès's approach. Travé-Massuyès *et al.,* (2006) proposed a method for finding all the ARRs in a physical system. This method was designed for diagnosability assessment. It starts from a structural model augmented by causal information. The starting point of this method is to assume that all variables of a system are measured. It proceeds by consecutively removing the hypothetical sensors. The successive removal of sensors is equivalent to successively combining ARRs.

This method computes all the possible causal interpretations of the system relations, and therefore all the ARRs can be found. The support of an ARR ($Supp(ARR)$) consists of a components set and a sensor set (i.e., a set of elementary constraints modelling a component set and a set of terminal constraints modelling a sensor set).

Consider the counter where the variables are $\{x_0, x_1, x_2, x_3, x_4\}$ and the data flows are $\{\tilde{x}_0, \tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4\}$. When all the variables are measured, ARRs are simply given by the primary relations in which every variable is replaced by its data flow.

$ARR_1 : \tilde{x_1} = \tilde{x_0} + 1, Supp(ARR_1) = \{\mathcal{K}_1, \mathcal{K}_5, \mathcal{K}_6\},$
$ARR_2 : \tilde{x_2} = \tilde{x_1} + 1, Supp(ARR_2) = \{\mathcal{K}_2, \mathcal{K}_6, \mathcal{K}_7\},$
$ARR_3 : \tilde{x_3} = \tilde{x_2} + 1, Supp(ARR_3) = \{\mathcal{K}_3, \mathcal{K}_7, \mathcal{K}_8\},$
$ARR_4 : \tilde{x_4} = \tilde{x_3} + 1, Supp(ARR_4) = \{\mathcal{K}_4, \mathcal{K}_8, \mathcal{K}_9\}.$

These ARRs can be interpreted causally. It yields

$ARR_1 : \tilde{x_1} = \tilde{x_0} + 1, Supp(ARR_1) = \{\mathcal{K}_1, \mathcal{K}_5, \mathcal{K}_6\},$
$ARR_2 : \tilde{x_0} = -\tilde{x_1} + 1, Supp(ARR_2) = \{\mathcal{K}_1, \mathcal{K}_5, \mathcal{K}_6\},$
$ARR_3 : \tilde{x_2} = \tilde{x_1} + 1, Supp(ARR_3) = \{\mathcal{K}_2, \mathcal{K}_6, \mathcal{K}_7\},$
$ARR_4 : \tilde{x_1} = -\tilde{x_2} + 1, Supp(ARR_4) = \{\mathcal{K}_2, \mathcal{K}_6, \mathcal{K}_7\},$
$ARR_5 : \tilde{x_3} = \tilde{x_2} + 1, Supp(ARR_5) = \{\mathcal{K}_3, \mathcal{K}_7, \mathcal{K}_8\},$
$ARR_6 : \tilde{x_2} = -\tilde{x_2} + 1, Supp(ARR_6) = \{\mathcal{K}_3, \mathcal{K}_7, \mathcal{K}_8\},$
$ARR_7 : \tilde{x_4} = \tilde{x_3} + 1, Supp(ARR_7) = \{\mathcal{K}_4, \mathcal{K}_8, \mathcal{K}_9\},$
$ARR_8 : \tilde{x_3} = -\tilde{x_4} + 1, Supp(ARR_8) = \{\mathcal{K}_4, \mathcal{K}_8, \mathcal{K}_9\}.$

In order to find all the ARRs, the successive removal of sensors (terminal constraints modelling these sensors) is applied. In this example, there are 32 possible removals. Consider that the sensor measuring the variable x_1 is removed, one new ARR is obtained by combining one ARR from $\{ARR_1, ARR_2\}$ with one ARR from $\{ARR_3, ARR_4\}$. By combining ARR_1 with ARR_3, a new ARR is obtained: $ARR_9 : \tilde{x_2} = \tilde{x_0} + 2$, $Supp(ARR_9) = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_5, \mathcal{K}_7\}$.

This method makes it possible to find all the ARRs of a system. The main disadvantage of this method is that the complexity is exponential according to variables. Moreover, this complexity increases taking into account the causal interpretation.

5.3. Krysander's approach. Krysander *et al.* (2008) proposed an 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 reduces the size of the model step by step until a TSS remains.

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

Given a bipartite graph $G = (\mathbb{K}, var(\mathbb{K}), \mathcal{E})$, where $\mathbb{K}$ is the set of constraints of a system, $var(K)$ is the subset of variables connected to at least one constraint in $\mathbb{K}$ and $\mathcal{E} = \mathbb{K} \times var(\mathbb{K})$ stands for edges in the graph, the structural redundancy φK is defined by $\varphi K = |K^+| - |var(K^+)|$, 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.

A digital system is used to illustrate this method. This system may be represented by the structural matrix given in Table 1, where $\mathbb{K} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_4, \mathcal{K}_5, \mathcal{K}_6, \mathcal{K}_7, \mathcal{K}_8, \mathcal{K}_9\}$ is the constraint set of the system. The structural redundancy of the system is 4.

Table 1. Structural matrix of a system.

	x_0	x_1	x_2	x_3	x_4
$\mathcal{K}_1$	1	1	0	0	0
$\mathcal{K}_2$	0	1	1	0	0
$\mathcal{K}_3$	0	0	1	1	0
$\mathcal{K}_4$	0	0	0	1	1
$\mathcal{K}_5$	1	0	0	0	0
$\mathcal{K}_6$	0	1	0	0	0
$\mathcal{K}_7$	0	0	1	0	0
$\mathcal{K}_8$	0	0	0	1	0
$\mathcal{K}_9$	0	0	0	0	1

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

- If $\mathcal{K}_1$ is removed, the Dulmage-Mendelshon decomposition of $\mathbb{K} \setminus \{\mathcal{K}_1\}$ is $(\mathbb{K} \setminus \{\mathcal{K}_1\})^+ = \{\mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_4, \mathcal{K}_6, \mathcal{K}_7, \mathcal{K}_8\}$, $(\mathbb{K}\setminus\{\mathcal{K}_1\})^0 = \{\mathcal{K}_5\}$ and $(\mathbb{K}\setminus\{\mathcal{K}_1\})^- = \{\emptyset\}$. The set $\mathbb{K}^{'} = (\mathbb{K}\setminus\{\mathcal{K}_1\})^+$ does not verify $\varphi\mathbb{K}^{'} = 1$; then the set $\mathbb{K}^{'}$ is not a TSS.

- If $\mathcal{K}_1, \mathcal{K}_2$ are removed, the Dulmage-Mendelshon decomposition of $\mathbb{K}\setminus\{\mathcal{K}_1, \mathcal{K}_2\}$ is $(\mathbb{K}\setminus\{\mathcal{K}_1, \mathcal{K}_2\})^+ =$

$\{\mathcal{K}_3, \mathcal{K}_4, \mathcal{K}_7, \mathcal{K}_8\}$, $(\mathbb{K} \setminus \{\mathcal{K}_1, \mathcal{K}_2\})^0 = \{\mathcal{K}_5, \mathcal{K}_6\}$ and $(\mathbb{K} \setminus \{\mathcal{K}_1, \mathcal{K}_2\})^- = \{\emptyset\}$. The set $\mathbb{K}^{'} = (\mathbb{K} \setminus \{\mathcal{K}_1, \mathcal{K}_2\})^+$ does not verify $\varphi\mathbb{K}^{'} = 1$; then the set $\mathbb{K}^{'}$ is not a TSS.

- If $\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3$ are removed, the Dulmage-Mendelshon decomposition of $\mathbb{K} \setminus \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3\}$ is

$$(\mathbb{K} \setminus \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3\})^+ = \{\mathcal{K}_4, \mathcal{K}_8, \mathcal{K}_9\},$$
$$(\mathbb{K} \setminus \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3\})^0 = \{\mathcal{K}_5, \mathcal{K}_6, \mathcal{K}_7\},$$
$$(\mathbb{K} \setminus \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3\})^- = \{\emptyset\}.$$

  The set $\mathbb{K}^{'} = (\mathbb{K} \setminus \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3\})^+$ satisfies $\varphi\mathbb{K}^{'} = 1$; then the set $\mathbb{K}^{'} = \{\mathcal{K}_4, \mathcal{K}_8, \mathcal{K}_9\}$ is a TSS.

In order to find all the testable subsystems, all the combinations of elimination must be made. In this example, 119 combinations were achieved in oder to find the ten possible TSSs of the system.

The same TSS can be found more than once. Accordingly, 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 cannot take into account the notion of deductibility of variables—all variables of the system are treated as deductibles and, therefore, some ARRs may not be achievable. Of course, systems with branchings cannot be managed.

5.4. Assessment of the proposed approach. The method proposed in Section 4 will now be applied. Because no structure has to be cleared, it starts with the definition of the set F_1, which is composed of the following formulas:

f_1 with $s(f_1) = \llcorner\{x_0, x_1\}\lrcorner$,
f_2 with $s(f_2) = \llcorner\{x_1, x_2\}\lrcorner$,
f_3 with $s(f_3) = \llcorner\{x_2, x_3\}\lrcorner$,
f_4 with $s(f_4) = \llcorner\{x_3, x_4\}\lrcorner$,
f_5 with $s(f_5) = \llcorner\{x_0\}\lrcorner$,
f_6 with $s(f_6) = \llcorner\{x_1\}\lrcorner$,
f_7 with $s(f_7) = \llcorner\{x_2\}\lrcorner$,
f_8 with $s(f_8) = \llcorner\{x_3\}\lrcorner$,
f_9 with $s(f_9) = \llcorner\{x_4\}\lrcorner$.

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 formulas where it applies. Then, removing the formulas containing the variable x_0, the following set of formulas F is obtained:

$f_1 \bowtie_{x_0} f_5$ with $s(f_1 \bowtie_{x_0} f_5) = \llcorner\{x_1\}\lrcorner$,
f_2 with $s(f_2) = \llcorner\{x_1, x_2\}\lrcorner$,
f_3 with $s(f_3) = \llcorner\{x_2, x_3\}\lrcorner$,
f_4 with $s(f_4) = \llcorner\{x_3, x_4\}\lrcorner$,
f_6 with $s(f_6) = \llcorner\{x_1\}\lrcorner$,
f_7 with $s(f_7) = \llcorner\{x_2\}\lrcorner$,
f_8 with $s(f_8) = \llcorner\{x_3\}\lrcorner$,
f_9 with $s(f_9) = \llcorner\{x_4\}\lrcorner$.

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

$f_1 \bowtie_{x_0} f_5$ with $s(f_1 \bowtie_{x_0} f_5) = \llcorner\{x_1\}\lrcorner$,
f_2 with $s(f_2) = \llcorner\{x_1, x_2\}\lrcorner$,
f_3 with $s(f_3) = \llcorner\{x_2, x_3\}\lrcorner$,
$f_4 \bowtie_{x_4} f_9$ with $s(f_4 \bowtie_{x_4} f_9) = \llcorner\{x_3\}\lrcorner$,
f_6 with $s(f_6) = \llcorner\{x_1\}\lrcorner$,
f_7 with $s(f_7) = \llcorner\{x_2\}\lrcorner$,
f_8 with $s(f_8) = \llcorner\{x_3\}\lrcorner$.

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 formulas where it applies. Then, removing the formulas containing the variable x_1, the following set of formulas F is obtained:

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

The 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 formulas where it applies. Then, removing the formulas containing the variable x_3, the following set of formulas F is obtained:

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

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

formulas containing the variable x_2, the resulting structures are

- $(f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6$ with $s\left((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6\right) = \llcorner\emptyset\lrcorner$,
- $(f_4 \bowtie_{x_4} f_9) \bowtie_{x_3} f_8$ with $s((f_4 \bowtie_{x_4} f_9) \bowtie_{x_3} f_8) = \llcorner\emptyset\lrcorner$,
- $f_7 \bowtie_{x_2} ((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2)$ with $s\left(f_7 \bowtie_{x_2} ((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2)\right) = \llcorner\emptyset\lrcorner$,
- $f_7 \bowtie_{x_2} (f_2 \bowtie_{x_1} f_6)$ with $s\left(f_7 \bowtie_{x_2} (f_2 \bowtie_{x_1} f_6)\right) = \llcorner\emptyset\lrcorner$,
- $f_7 \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))$ with $s\left(f_7 \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))\right) = \llcorner\emptyset\lrcorner$,
- $f_7 \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)$ with $s\left(f_7 \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)\right) = \llcorner\emptyset\lrcorner$,
- $((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))$ with $s(((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))) = \llcorner\emptyset\lrcorner$,
- $((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)$ with $s\left(((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)\right) = \llcorner\emptyset\lrcorner$,
- $(f_2 \bowtie_{x_1} f_6) \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))$ with $s\left((f_2 \bowtie_{x_1} f_6) \bowtie_{x_2} (f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9))\right) = \llcorner\emptyset\lrcorner$,
- $(f_2 \bowtie_{x_1} f_6) \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)$ with $s\left((f_2 \bowtie_{x_1} f_6) \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)\right) = \llcorner\emptyset\lrcorner$.

The formulas $((f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_2) \bowtie_{x_2} (f_2 \bowtie_{x_1} f_6)$ and $(f_3 \bowtie_{x_3} (f_4 \bowtie_{x_4} f_9)) \bowtie_{x_2} (f_3 \bowtie_{x_3} f_8)$ were calculated and then removed because the formulas $(f_1 \bowtie_{x_0} f_5) \bowtie_{x_1} f_6$ and $(f_4 \bowtie_{x_4} f_9) \bowtie_{x_3} f_8$ are respectively simpler. Only 17 elementary propagations were necessary to find all the MPTFs instead of 256 ways of propagations with the Bipartite graph approach, 119 eliminations with Krysander's approach, and 32 eliminations (without taking into account the causal interpretation) with the Travé-Massuyès' approach.

6. Application example

The algorithm presented in Section 4 has been implemented as a specialized language interpreter[1] in order to facilitate the search for all the possible tests that can be performed in a system.

6.1. System description. Consider the following electronic circuit in Fig. 6. Because of the number of observations, this example is so complex that it is hard to solve manually.

[1]The software can be obtained from the authors.

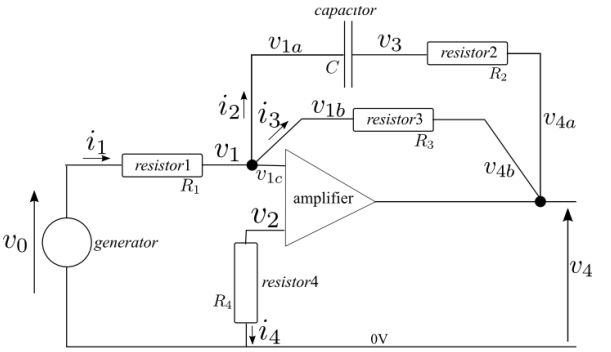

Fig. 6. Diagram of an electronic circuit.

There are four voltage sensors measuring v_1, v_2, v_3 and v_4, and one voltage generator applying the voltage v_0. The system is modeled by the following constraints:

$$
\begin{aligned}
amplifier &: k_1 : v_{1c} = v_2,\\
connection1 &: k_2 : i_1 = i_2 + i_3,\\
connection1 &: k_3 : v_1 = v_{1a},\\
connection1 &: k_4 : v_1 = v_{1b},\\
connection1 &: k_5 : v_1 = v_{1c},\\
resistor1 &: k_6 : v_0 - v_1 = R_1 i_1,\\
capacitor &: k_7 : C(v_{1a} - v_3) = \int_0^t i_2 \,\mathrm{d}t,\\
resistor2 &: k_8 : v_3 - v_{4a} = R_2 i_2,\\
resistor3 &: k_9 : v_1 - v_{4b} = R_3 i_3,\\
resistor4 &: k_{10} : v_2 = R_4 i_4,\\
connection2 &: k_{11} : v_4 = v_{4a},\\
connection2 &: k_{12} : v_4 = v_{4b},\\
generator &: k_{13} : \tilde{v_0} = v_0,\\
sensor1 &: k_{14} : \tilde{v_1} = v_1,\\
sensor2 &: k_{15} : \tilde{v_2} = v_2,\\
sensor3 &: k_{16} : \tilde{v_3} = v_3,\\
sensor4 &: k_{11} : \tilde{v_4} = v_4.
\end{aligned}
$$

6.2. Solutions and discussion. According to Section 4.4.1, the capacitor's constraint is structurally modeled. The following formulas arise:

- amplifier:f_1 with $s(f_1) = \llcorner\{v_{1c}, v_2\}\lrcorner$,
- connection1f_2 with $s(f_2) = \llcorner\{i_1, i_2, i_3\}\lrcorner$,
- connection1:f_3 with $s(f_3) = \llcorner\{v_1, v_{1a}\}\lrcorner$,
- connection1:f_4 with $s(f_4) = \llcorner\{v_1, v_{1b}\}\lrcorner$,
- connection1: f_5 with $s(f_5) = \llcorner\{v_1, v_{1c}\}\lrcorner$,
- resistor1: f_6 with $s(f_6) = \llcorner\{v_0, v_1, i_1\}\lrcorner$,

- capacitor:f_7 with $s(f_7) = \llcorner\{v_{1a}, v_3\}, \{i_2\}\lrcorner$,
- resistor2:f_8 with $s(f_8) = \llcorner\{v_3, v_{4a}, i_2\}\lrcorner$,
- resistor3: f_9 with $s(f_9) = \llcorner\{v_1, v_{4b}, i_3\}\lrcorner$,
- resistor4:f_{10} with $s(f_{10}) = \llcorner\{v_2, i_4\}\lrcorner$,
- connection2:f_{11} with $s(f_{11}) = \llcorner\{v_4, v_{4a}\}\lrcorner$,
- connection2:f_{12} with $s(f_{12}) = \llcorner\{v_4, v_{4b}\}\lrcorner$,
- generator:f_{13} with $s(f_{13})\llcorner\{v_0\}\lrcorner$,
- sensor1:f_{14} with $s(f_{14})\llcorner\{v_1\}\lrcorner$,
- sensor2:f_{15} with $s(f_{15}) = \llcorner\{v_2\}\lrcorner$,
- sensor3:f_{16} with $s(f_{16}) = \llcorner\{v_3\}\lrcorner$,
- sensor4:f_{17} with $s(f_{17}) = \llcorner\{v_4\}\lrcorner$.

Note that for the capacitor only the variable i_2 is considered deductible in order to avoid derivations that amplify high frequency noises. Then, the following test formulas are obtained thanks to the proposed method:

- $t_1 = (f_{14} \bowtie_{v_1} (f_{16} \bowtie_{v_3} (f_{17} \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} f_7) \bowtie_{i_2} ((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13})))))$,
- $t_2 = ((f_{16} \bowtie_{v_3} (f_{17} \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} f_7) \bowtie_{i_2} ((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f12) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13})))) \bowtie_{v_1} (f_{16} \bowtie_{v_3} (f_{17} \bowtie_{v_4} (((((f_2 \bowtie_{i_3} f_9) \bowtie_{v4b} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13}) \bowtie_{i_2} (f_{11} \bowtie_{v4a} f_8)))))$,
- $t_3 = (f_{14} \bowtie_{v_1} (f_{16} \bowtie_{v_3} (f_{17} \bowtie_{v_4} (((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13}) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8)))))$,
- $t_4 = ((f_{15} \bowtie_{v_2} (f_5 \bowtie_{v_{1c}} f_1)) \bowtie_{v_1} (f_{16} \bowtie_{v_3} (f_{17} \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} f_7) \bowtie_{i_2} ((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13})))))$,
- $t_5 = (((f_{17} \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} f_7) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8))) \bowtie_{v_3} f_{16}) \bowtie_{v_1} f_{14})$,
- $t_6 = ((f_{15} \bowtie_{v_2} (f_5 \bowtie_{v_{1c}} f1)) \bowtie_{v_1} f_{14})$,
- $t_7 = (f_{14} \bowtie_{v_1} ((f_{17} \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} f_7) \bowtie_{i_2} ((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13}))) \bowtie_{v_3} (((((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13}) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8)) \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} f_7) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8)))))$,
- $t_8 = (((f_{17} \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} f_7) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8))) \bowtie_{v_3} f_{16}) \bowtie_{v_1} (f_{15} \bowtie_{v_2} (f_5 \bowtie_{v_{1c}} f_1)))$,
- $t_9 = (f_{14} \bowtie_{v_1} (f_{16} \bowtie_{v_3} (((((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13}) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8)) \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} s_7) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} s_8)))))$,
- $t_{10} = ((f_{15} \bowtie_{v_2} (f_5 \bowtie_{v_{1c}} f_1)) \bowtie_{v_1} ((f_{17} \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} f_7) \bowtie_{i_2} ((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13}))) \bowtie_{v_3} (((((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13}) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8)) \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} f_7) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8)))))$,
- $t_{11} = ((f_{15} \bowtie_{v_2} (f_5 \bowtie_{v_{1c}} f_1)) \bowtie_{v_1} (f_{16} \bowtie_{v_3} (f_{17} \bowtie_{v_4} (((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13}) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8)))))$,
- $t_{12} = ((f_{15} \bowtie_{v_2} (f_5 \bowtie_{v_{1c}} f_1)) \bowtie_{v_1} (f_{16} \bowtie_{v_3} (((((((f_2 \bowtie_{i_3} f_9) \bowtie_{v_{4b}} f_{12}) \bowtie_{i_1} f_6) \bowtie_{v_0} f_{13}) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8)) \bowtie_{v_4} ((f_3 \bowtie_{v_{1a}} f_7) \bowtie_{i_2} (f_{11} \bowtie_{v_{4a}} f_8)))))$.

In gathering the structures modeling the same components, the checked components can be deduced:

- t_1: connection2, connection1, resistor2, amplifier, sensor4, capacitor, sensor3, sensor2,
- t_2: connection1, amplifier, sensor2, sensor1,
- t_3: amplifier, generator, resistor3, capacitor, resistor2, resistor1, connection2, connection1, sensor4, sensor2,
- t_4: sensor1, generator, resistor3, resistor2, resistor1, connection2, connection1, sensor4, sensor3,
- t_5: sensor1, generator, resistor3, capacitor, resistor2, resistor1, connection2, connection1, sensor4,
- t_6: sensor1, generator, resistor3, capacitor, resistor2, resistor1, connection2, connection1, sensor3,
- t_7: amplifier, generator, resistor3, capacitor, resistor2, resistor1, connection2, connection1, sensor3, sensor2,
- t_8: amplifier, generator, resistor3, resistor2, resistor1, connection2, connection1, sensor4, sensor3, sensor2,
- t_9: connection2, resistor2, connection1, capacitor, sensor4, sensor3, sensor1,
- t_{10}: amplifier, generator, resistor3, capacitor, resistor1, connection2, connection1, sensor4, sensor3, sensor2,
- t_{11}: generator, resistor3, capacitor, resistor2, resistor1, connection2, connection1, sensor4, sensor3,
- t_{12}: sensor1, generator, resistor3, capacitor, resistor1, connection2, connection1, sensor4, sensor3.

A signature table, where rows correspond to tests and columns to components, summarized the results. In studying the signature of each component, it is possible to study diagnosability (see (Console *et al.*, 2000) for a definition).

Detectability, i.e., the possibility of detecting a fault on a component, and diagnosability, i.e., the possibility of isolating a fault on a component, can now be studied thanks to the signature table (Travé-Massuyès *et al*., 2003). Because all the ARRs have been found, it is easy to determine the best properties for this system:

- Diagnosable components are: connection2, connection1, resistor2, sensor4, capacitor, sensor3, sensor1;
- Detectable but not discriminable components are:
  - amplifier, sensor2,
  - generator, resistor3, resistor1.

Detection tests can then be designed using various tools including state observers or parity relations (see (Staroswiecki *et al*., 1991) for an example) for constraints containing ordinary differential equations. Consider, for instance, the formula t_9. It may yield the state observer:

$$\begin{cases} C\dfrac{\mathrm{d}x}{\mathrm{d}t} = -\dfrac{1}{R_2+R_3}x + \dfrac{R_3}{R_1(R_2+R_3)}(\tilde{v_0}-\tilde{v_1}) \\ \qquad\qquad +K\left(x-\tilde{v_1}-\tilde{v_3}\right), \\ \tilde{v_1}-\tilde{v_3}=x. \end{cases}$$

The gain matrix can be adjusted to improved the detection capabilities in the presence of noise.

7. Conclusion

This paper formalizes structural modeling and shows how it can support engineers in constructing detection tests for the system to be diagnosed. The proposed methods provide the constraints to be used to design each test, but also a way of combining one with another. It then lets the engineer choose its preferred detection test (for instance, parity relations, Luenberger state observers or Kalman filters in dynamic continuous time systems), the way of tuning the detection tests *a posteriori* in order to take into account modeling uncertainties.

The main advantage of structural modeling is that it makes it possible to handle any kind of systems, e.g., dynamic continuous-time, discrete event or rule-based systems. In the paper, it was shown on a road network that, even if the behavioral constraints are not available, it is still possible to compute the composition of all the possible ARRs.

The drawback of structural approaches is that overestimations of some solutions may occur. Even if this can be partially avoided in taking into account the realizability of the constraints, some over-estimations are still possible. A consequence is that some provided constraints in an ARR may not be required. They have to be removed afterwards when designing the detection tests. This drawback is not a major issue because the main problem that has to be tackled when designing tests in complex systems is to determine the sets of constraints that lead to an ARR.

The proposed procedure was designed in order to reduce as much as possible the number of computations. It was compared with two alternative approaches and significant improvements were pointed out: it requires fewer computations and handles deductibility and exclusions.

Because the constraints included in all the ARRs are provided, a complete signature table can be written. Therefore, it becomes possible to determine the best achievable performances of diagnostic procedures. Moreover, because the supports of tests are provided, the results are particularly suitable for bridge approaches to fault diagnosis.

References

Blanke, M., Kinnaert, M. and Staroswiecki, M. (2003). *Diagnosis and Fault Tolerant Control*, Springer, Berlin.

Cassar, J. and Staroswiecki, M. (1997). A structural approach for the design of failure detection and identification systems, *IFAC, IFIP, IMACS Conference on Control of Industrial Systems, Belfort, France*, pp. 329–334.

Chittaro, L. and Ranon, R. (2004). Hierarchical model-based diagnosis based on structural abstraction, *Artificial Intelligence* **155**(1-2): 147–182.

Codd, E. (1970). A relational model of data for large shared data banks, *Communications of the ACM* **13**(6): 377–387.

Console, L., Picardi, C. and Ribando, M. (2000). Diagnosis and diagnosability analysis using process algebra, *Proceedings of the Eleventh International Workshop on Principles of Diagnosis (DX-00), MX, Morelia, Mexico*, pp. 25–32.

Dague, P. (2001). Théorie logique du diagnostic à base de modèles, *in* B. Dubuisson (Ed.), *Diagnostic, Intelligence artificielle et reconnaissance de formes*, Hermès Science, Paris, pp. 17–104.

Davis, R. (1984). Diagnostic reasoning based on structure and behavior, *Artificial Intelligence* **24**(1–3): 347–410.

De Kleer, J. and Williams, B. C. (1987). Diagnosing multiple faults, *Artificial Intelligence* **32**(1): 97–130.

de Kleer, J. and Williams, B. C. (1992). Diagnosis with behavioral modes, *in* W.C. Hamscher, I.de Kleer and L. Console (Eds), *Readings in Model-Based Diagnosis*, Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 124–130.

Dechetr, R. (2003). *Constraint Processing*, Morgan Kaufmann Publishers, San Francisco, CA.

Declerck, P. and Staroswiecki, M. (1991). Characterization of the canonical components of a structural graph for fault detection in large scale industrial plants, *European Control Conference, Grenoble, France*, pp. 298–303.

Dulmage, A. L. and Mendelsohn, N. S. (1959). A structure theory of bi-partite graphs of finite exterior extension, *Transactions of the Royal Society of Canada* **53**(III): 1–13.

Frank, P. M. (1990). Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy—A survey and some new results, *Automatica* **26**(3): 459–471.

Frisk, E. (2000). Residual generator for non-linear polynomial systems—A Grobner basis approach, *IFAC Fault Detection, Supervision and Safety for Technical Processes, Budapest, Hungary*, pp. 979–984.

Fron, A. (1994). *Programmation par contraintes*, Addison-Wesley, Paris.

Górny, B. and Ligęza, A. (2001). Review of systematic conflict generation in model-based diagnosis of dynamic systems, *IFAC Workshop on Manufacturing, Modelling Manageent and Control, Prague, Czech Republic*, pp. 86–91.

Iwasaki, Y. and Simo, H. A. (1994). Causality and model abstraction, *Artificial Intelligence* **67**(1): 143–194.

Krysander, M., Åslund, J. and Nyberg, M. . (2008). 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.

Krysander, M., Aslund, J. and Nyberg, M. (2005). An efficient algorithm for finding over-constrained sub-systems for construction of diagnostic tests, *16th International Workshop on Principles of Diagnosis (DX-05), Pacific Grove, CA, USA*.

Ligęza, A. and Górny, B. (2000). Systematic conflict generation in model-based diagnosis, *SAFEPROCESS'2000: 4th IFAC Symposium on Fault Detection and Supervision and Safety for Technological Processes, Budapest, Hungary*, Vol. II, pp. 1103–1108.

Mishra, P. and Eich, M. (1992). Join processing in relational databases, *ACM Computing Surveys* **24**(1): 63–113.

Nayak, P. P. and Levy, A. Y. (1995). A semantic theory of abstractions, *14th International Joint Conference on Artificial Intelligence IJCAI-95, Montreal, Canada*, pp. 196–203.

Nyberg, M. and Krysander, M. (2003). Combining AI, FDI, and statistical hypothesis-testing in a framework for diagnosis, *IFAC SAFEPROCESS'03, Washington, DC, USA*, pp. 813–818.

Patton, R., Frank, P. and Clark (Eds), R. (1989). *Fault Diagnosis in Dynamic Systems*, International Series in Systems and Control Engineering, Prentice Hall, London.

Ploix, S., Désinde, M. and Michau, F. (2004). Assessment and diagnosis for virtual reality training, *International Symposium on Advanced Robot Systems and Virtual Reality, Grenoble, France.*

Ploix, S., Desinde, M. and Touaf, S. (2005). Automatic design of detection tests in complex dynamic systems, *16th IFAC World Congress, Prague, Czech Republic.*

Ploix, S., Touaf, S. and Flaus, J. M. (2003). A logical framework for isolation in fault diagnosis, *SAFEPROCESS'2003, Washington, DC, USA.*

Pulido, B. and Alonso, C. (2002). Possible conflicts, arrs, and conflicts, *13th International Workshop on Principles of Diagnosis (DX02), Semmering, Austria*, pp. 122–128.

Reiter, R. (1987). A theory of diagnosis from first principles, *Artificial Intelligence* **32**(1): 57–95.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence, A Modern Approach, 2nd Ed.*, Prentice Hall, Upper Saddle River, NJ.

Staroswiecki, M., Cocquempot, V. and Cassar, J. P. (1991). Observer based and parity space approaches for failure detection and identification, *IMACS-IFAC International Symposium, Lille, France*, Vol. 25, pp. 536–541.

Staroswiecki, M. and Declerck, P. (1989). Analytical redundancy in nonlinear interconnected systems by means of structural analysis, *IFAC AIPAC'89 Symposium, Nantes, France*, Vol. 2, pp. 23–27.

Struss, P. (1992). What's in SD? Towards a theory of modeling for diagnosis, *in* W. Hamscher, L. Console and J. De Kleer (Eds), *Readings in Model-Based Diagnosis*, Morgan Kaufman, San Francisco, CA, pp. 419–448.

Travé-Massuyès, L., Escobet, T. and Olive, X. (2006). 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.

Travé-Massuyès, L., Escobet, T. and Spanache, S. (2003). Diagnosability analysis based on component supported analytical redundancy relations, *IFAC Workshop SAFEPROCESS'2003, Washington, DC, USA*, pp.897–902.

Willsky, A. (1976). A survey of design methods for failure detection in dynamic systems, *Automatica* **21**(4): 601–611.

Stéphane Ploix is *Maître de Conférences* at the Grenoble Institute of Technology in the G-SCOP lab. After an engineer degree in mechanics and electricity, in 1998 he obtained a Ph.D. from *Institut National Polytechnique de Lorraine* in control engineering and signal processing. He is a specialist in supervision, monitoring and diagnosis, and his studies focus on human-machine cooperative mechanisms. He is involved in different industrial projects dealing with the supervision of distributed plants, the diagnosis of human skills, iterative diagnosis tool for companies and power management in buildings.

Abed Alrahim Yassine occupies a postdoctoral position at the Grenoble Institute of Technology in the G-SCOP lab. In 1998, he obtained an engineer degree in electronic engineering from Tishrine University, Faculty of Mechanical and Electrical Engineering, in Syria. In 2008, he obtained a Ph.D. from Joseph Fourrier University in France in automatic control and industrial automation. His research focuses on fault diagnosis of industrial plants. He has developed tools for the design of testable subsystems and sensor placement.

Jean-Marie Flaus received a Ph.D. degree in automatic control in 1990 from the Grenoble Institute of Technology, after obtaining an electrical engineering degree. He worked for three years for a large chemical company and then joined CNRS for six years. He is currently a full professor at Joseph Fourier University, where he teaches mainly process safety. He joined the G-SCOP lab in 2006. His research interests include risk analysis methods, diagnosis, system safety and interval analysis for the monitoring and control of complex systems.

Received: 23 October 2008
Revised: 24 July 2009