Sur le diagnostic interactif

Quang Huy Giap

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **AUTOMATIQUE - PRODUCTIQUE**

Arrêté ministériel : 7 août 2006

Présentée par

Quang Huy GIAP

Thèse dirigée par **Jean-Marie FLAUS** et
codirigée par **Stéphane PLOIX**

préparée au sein du **Laboratoire des Sciences pour la Conception, l'Optimisation et la Production de Grenoble (G-SCOP) - UMR5272**
dans **l'École Doctorale Electrique, Electronique, Automatique, Traitement du Signal (EEATS)**

Sur le diagnostic interactif

Thèse soutenue publiquement le **8 Juin 2011**,
devant le jury composé de :

Monsieur Jacky MONTMAIN
Professeur à l'Ecole des Mines d'Alès, Président
Madame Catherine TESSIER
Enseignante de rang équivalent à celui de Professeur des Universités, ONERA, Toulouse, Rapporteur
Monsieur Kondo ADJALLAH
Professeur à l'Ecole Nationale d'Ingénieurs de Metz, Rapporteur
Monsieur Jean-Baptiste LEGER
Docteur, président de PREDICT, Examinateur
Madame Maria DI MASCOLO
Chargé de recherche au CNRS, Examinatrice
Monsieur Jean Marie FLAUS
Professeur à l'UJF Grenoble, Directeur de thèse
Monsieur Stéphane PLOIX
Maître de conférences HDR à Grenoble-INP, Co-Directeur de thèse

Avant-Propos

Une thèse est rarement le travail d'une personne seule, mais elle est le résultat de rencontres, d'échanges et de discussions. Je tiens donc à remercier ici les personnes qui ont contribué à la concrétisation de ce travail de thèse de doctorat.

Je tiens à remercier tout d'abord mes directeurs de thèse : Monsieur Jean-Marie Flaus et Monsieur Stéphane Ploix qui m'ont proposé un sujet et qui m'ont encadré tout au long de ces années d'étude. Malgré de nombreuses sollicitations, ils m'ont toujours consacré du temps et m'ont donné l'assistance nécessaire. Un grand merci à Monsieur Jean Marie Flaus qui m'a guidé dans l'analyse et le traitement rigoureux des étapes successives de recherche et des problèmes rencontrées. Je souhaite remercier particulièrement Monsieur Stéphane Ploix qui m'a soutenu chaleureusement et m'a donné un regard scientifique et pratique permettant de mieux comprendre le sujet dans le contexte industriel. Par sa diligence et aussi son humanité, il m'a toujours aidé aussi bien dans la vie professionnelle que privée.

J'aimerais aussi remercier Monsieur Yannick Frein qui m'a accueilli au sein du laboratoire G-SCOP dans le cadre de cette thèse de doctorat. Je remercie aussi tous les membres de l'équipe GCSP pour leur accueil et leur soutien.

Je souhaite remercier Monsieur Jacky Montmain, Professeur à l'Ecole des Mines d'Alès, qui m'a fait l'honneur de présider le jury de cette thèse.

Je tiens à exprimer ma profonde gratitude à Madame Catherine Tessier, enseignante de rang equivalent à celui de Professeur des Universités à ONERA, Toulouse, et à Monsieur Kondo Hloindo Adjallah, Professeur à l'Ecole Nationale d'Ingénieurs de Metz, d'avoir accepté le rôle de rapporteur de ma thèse. Leurs commentaires et leurs questions m'ont permis d'améliorer de manière significative le document.

Ma gratitude s'adresse aussi à Madame Maria Di Mascolo, Chargé de recherche au CNRS et à Monsieur Jean-Baptiste Leger, Président de PREDICT, d'avoir accepté de prendre ce travail en considération en tant qu'examinateurs de ce jury.

Je remercie Monsieur Christian Depraetere et Monsieur Olivier Adrot, qui ont contribué à ma compréhension des problèmes liés au logiciel hydrodiag. Je leur suis reconnaissant également pour leur disponibilité et toute l'aide précieuse qu'ils m'ont apporté.

La rédaction est aussi une tâche important dans mes travaux de thèse. Mes remerciements s'adressent aussi à Madame Catherine Mucha-Ploix, qui m'a aidé à améliorer l'orthographe de mon manuscrit.

Pour terminer, je tiens à exprimer tous mes remerciements à mes parents et à ma famille entière qui m'ont toujours soutenu tout au long de ma carrière. A eux je dédie cette thèse.

Table des matières

Introduction

De nos jours, les systèmes industriels deviennent de plus en plus complexes. Dans ce contexte, la sécurité des installations, la maintenance des systèmes et des méthodes de diagnostic permettant de déterminer les modes de défauts d'un système quand il est en panne présentent un enjeu très important. Les outils de maintenance doivent permettre de surveiller le système, de lancer des alarmes quand les dysfonctionnements apparaissent, de proposer des diagnostics fiables afin de minimiser le temps de localisation et de préparation des défauts.

Un problème de diagnostic pour un système concret dépend de la pertinence de l'information recueillie sur le système. Généralement, la conception d'un problème de diagnostic comporte trois étapes principales : la modélisation du système, la détection et la localisation des défauts. Les informations peuvent être exploitées de façon implicite ou explicite dans ces trois étapes.

Une partie de l'expertise peut être explicitée sous forme de modèle du système. Pour les systèmes complexes, la difficulté de la tâche de modélisation se rapporte au niveau de détail qu'on veut analyser. La modélisation détaillée et complète pour un système est parfois coûteuse et prend beaucoup de temps. Surtout, il n'est pas réaliste pour un problème de diagnostic dit juste à temps, par exemple du travail des agences de maintenances industrielles. De cette limite, la construction au fur et à mesure du modèle du système aux différents niveaux d'abstraction durant le processus de diagnostic est une problématique de recherche pour notre travail.

Il peut exister certains autres types d'expertise qui ne sont pas explicités sous forme du modèle formel qui pourtant peuvent être exploités pour affiner les résultats de diagnostic. La question proposée est : comment exploiter l'expertise implicite durant le processus d'analyse diagnostique ?

Le travail que nous présentons dans ce mémoire propose d'apporter des contributions en distinguant trois problèmes de diagnostic itératif :

- le diagnostic itératif basé sur un modèle structuro-fonctionnel
- le diagnostic itératif basé sur un modèle comportemental avec propagation de valeurs
- le diagnostic itératif basé sur la connaissance experte implicite durant la phase d'analyse diagnostique

Les deux premiers problèmes sont liés à la complexité de la tâche de modélisation. Au lieu de construire un modèle complet et détaillé du modèle avant d'initier l'analyse diagnostique, l'analyse diagnostique est considérée comme un processus où le système est décomposé et son modèle est construit au fur et à mesure. Selon les symptômes observés, certaines parties du modèle seront affinées. Pour atteindre cet objectif, une approche de modélisation structuro-fonctionnelle qui ne demande pas beaucoup de connaissance explicite est proposée pour résoudre le premier problème. Elle permet à l'expert de construire le modèle de façon simple et d'expliciter certaines connaissances sur le système. En comparaison avec le premier problème, la complexité de la modélisation considérée dans le deuxième problème ne repose pas seulement sur la structure du système (le nombre d'éléments impliqués dans le système) mais aussi sur le comportement du système. Le modèle comportemental est donc intégré dans ce deuxième problème pour décrire le comportement du système.

Le troisième problème exprime le cas où l'expertise non-formulée est exploitée dans la phase de diagnostic pour affiner les résultats de diagnostic. Dans ce problème, le modèle du système est établi au début. Le modèle hiérarchique du système n'est pas considéré dans ce cas.

Ce mémoire est organisé en 6 chapitres dont les thèmes sont donnés ci-après :

Le premier chapitre présente de manière générale le contexte du problème et introduit les trois problèmes à résoudre dans les chapitres suivants. Une étude de bibliographie est réalisée afin de positionner les travaux de ce manuscrit. Ce positionnement repose sur le choix de l'approche de diagnostic, sur le choix de l'approche de modélisation hiérarchique et sur la façon dont nous exploitons l'interaction homme-automate dans l'analyse diagnostique.

Le deuxième chapitre est consacré à la présentation des éléments de formulation et aux notions fondamentales que nous voulons étudier. Tout d'abord, les éléments de formulation de base pour l'approche hybride de diagnostic DX-FDI sont rappelés. En parallèle, les nouveaux éléments de formulation sont présentés en visant la construction d'un modèle d'abstraction. Ensuite, un modèle hiérarchique est défini via les éléments de base du système : variable, modèle élémentaire et item (fonction ou ressource). Cette partie donne une vue globale sur la modélisation que nous allons utiliser dans les problèmes de diagnostics itératifs.

Le troisième chapitre présente un processus de diagnostic avec interactions homme-automate qui est basé sur le modèle structuro-fonctionnel FIS. Ceci permet de résoudre les difficultés qui se rapportent à la complexité de la structure du système (le nombre important d'éléments impliqués dans le système). Ce problème concerne le travail des services de maintenance. Pour aider l'expert et pour faire face à la variété des systèmes, une démarche de diagnostic qui ne requiert pas de formulation détaillée et complète du système au préalable a été proposée. Le problème de diagnostic est présenté comme un processus interactif homme-automate, où le modèle du système est découvert et explicité au fur et à mesure de l'avancement du processus. Un outil d'aide au diagnostic a été conçu et permet de calculer des

diagnostics à partir du modèle structuro-fonctionnel décrit aux différents niveaux de détail et de guider l'expert dans son analyse et ses observations.

Dans le quatrième chapitre, un processus de diagnostic est présenté avec interactions homme-automate en phase de modélisation. Cette modélisation comporte la modélisation structuro-fonctionnelle et la modélisation comportementale du système. En comparaison avec le problème présenté dans le troisième chapitre, la complexité du modèle exprimée dans ce problème repose non seulement sur la structure du système (l'ensemble important des éléments impliqués dans le système) mais aussi sur le comportement des items. La modélisation structuro-fonctionnelle est donc réalisée en parallèle avec la modélisation comportementale pour décrire un système au fur et à mesure aux différents niveaux d'abstraction dans un processus de diagnostic. A partir du modèle hiérarchique et des observations, une approche est proposée pour collecter automatiquement des symptômes. Cette approche doit collecter les symptômes qui existent pour calculer des diagnostics et éviter des symptômes fallacieux causés par certaines relations d'abstraction entre les items. Une fois que les symptômes sont collectés, les diagnostics peuvent être calculés par les approches proposées dans le chapitre 3.

Le cinquième chapitre présente un processus de diagnostic avec interaction homme-automate dans la phase de diagnostic pour exploiter l'expertise implicite. Le cas d'étude présenté est le problème de détection de défaut sur un réseau de pluviomètres distribués sur un bassin versant. Les tests résultent de corrélations entre les flux de données issus des capteurs. Tous les diagnostics possibles sont calculés à partir des tests négatifs. Avec une connaissance tacite complémentaire, un expert peut affiner le résultat de diagnostic à partir d'un ensemble important de propositions de l'automate. La logique floue a été utilisée en complément de la logique nette (négatif, positif) pour offrir plus de souplesse aux experts.

Chapitre 1

Contexte général

1.1 Introduction

Dans la littérature scientifique, la plupart des travaux publiés cherchent à automatiser les procédures de diagnostics en supposant que le modèle du système est complètement connu lorsque l'analyse diagnostique est initialisée. En réalité, cette supposition n'est pas toujours vraie, ce problème avait été discuté dans (Ploix et Chazot [2006],Giap *et al.* [2009]) et a été rappelé dans le contexte général de ce mémoire. Il peut être résumé en deux points principaux :

- le modèle du système est décrit avec différents niveaux de détail. Il est trop difficile de modéliser complètement le système avant l'analyse diagnostique. Donc, le diagnostic devient un processus où le système est décrit partiellement et certaines parties seront affinées au fur et à mesure.
- En plus du modèle formalisé, il existe d'autres types d'expertises non-formalisées qui sont implicites dans la connaissance de l'expert qui se construit au fur et à mesure de la procédure de diagnostic.

A partir d'une vue générale de l'objectif à atteindre, une étude de l'état de l'art a été réalisée pour permettre de positionner notre problématique dans la littérature existante d'une part et de choisir l'approche qui convient mieux au contexte considéré d'autre part. Ce positionnement repose sur les trois questions suivantes : Quelle approche de diagnostic sera choisie ? Quelle approche de modélisation permettra de répondre à nos besoins ? Comment l'interaction homme-automate permet d'exploiter les connaissances implicites de l'expert dans l'analyse diagnostique ?

1.2 Méthodes de diagnostic

1.2.1 Méthode de diagnostic à base d'arbre de décision

L'arbre de décision (Quinlan [1986], Price [1999]) est un outil qui permet de créer des scénarios pour diagnostiquer un système. Chaque scénario est décrit sous forme arborescente comportant des noeuds et des branches descendantes de chaque noeud. Chaque noeud est étiqueté par une question ou une décision. Un arbre de décision commence par un noeud initial nommé "racine" et il se termine par des noeuds terminaux nommés "feuilles". La racine et les noeuds intermédiaires guident l'opérateur à effectuer des tests ou des mesures sur le système. Les feuilles donnent des diagnostics finaux. Selon la réponse à la question, une branche descendante de l'arbre est atteinte. Nous prenons ici l'exemple de l'arbre de décision pour diagnostiquer une voiture qui ne peut pas être démarrée (Price [1999])(figure 1.1).

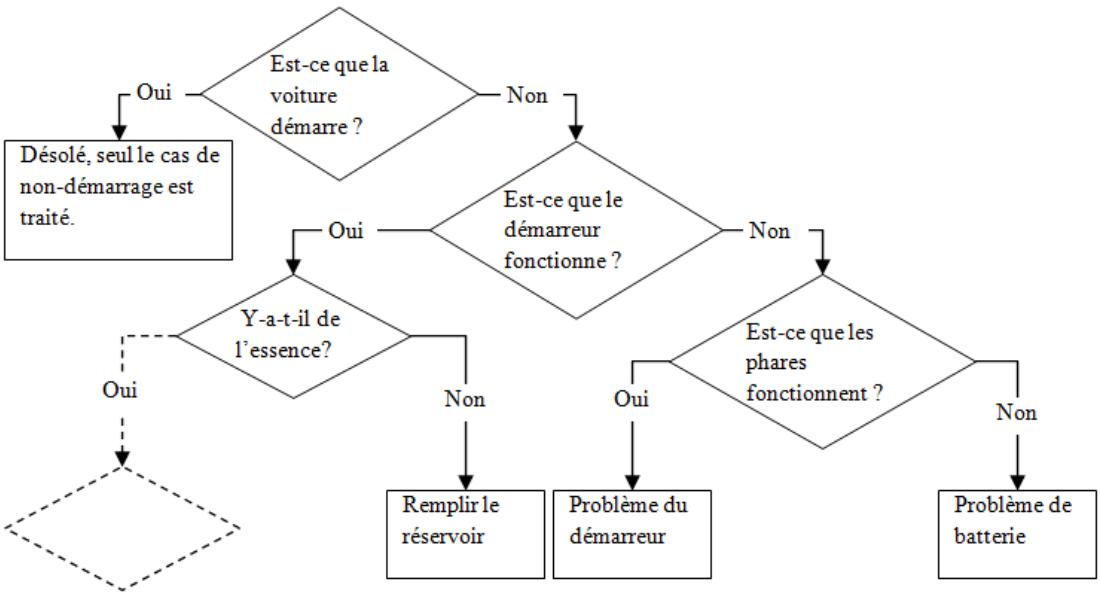


FIG. 1.1 – Diagnostic à base de l'arbre de décision

Certains travaux ont perfectionné ce principe de diagnostic. Ils permettent par exemple d'analyser des systèmes très complexes ou de faire une analyse croisée de plusieurs arbres pour résoudre un même problème.

L'avantage de l'arbre de décision est qu'il donne un outil simple et facile à appréhender pour décrire des procédures de résolution de problème. Il peut être utilisé pour les systèmes de diverses natures. Potentiellement, il permet de diagnostiquer des défauts multiples en garantissant les diagnostics obtenus mais avec condition que toutes les situations soient présentées.

L'inconvénient de cette approche est qu'elle dépend totalement de l'expertise et de la diversité des situations d'observation. Pour des symptômes initiaux différents, les arbres peuvent

être différents.

1.2.2 Analyse diagnostique à base de reconnaissance de cas

Le raisonnement à base de cas ou CBR (case-based reasoning) est une méthode qui utilise des solutions antérieures à des cas particuliers de problème pour résoudre le cas similaire (Kolodner [1993], Watson [1997], Price [1999]). Dans ce but, les effets observés des défauts qui se sont produits par le passé et les solutions correspondantes sont enregistrés dans une base de connaissances. Puis, lorsque des faits anormaux se produisent il s'agit de rechercher des cas similaires dans la base de connaissances pour trouver des diagnostics possibles. Si le problème a des aspects nouveaux, alors la solution du nouveau problème sera adaptée, évaluée et ajoutée à la base de données. Le déroulement de cette approche est présenté sur la figure (1.2).

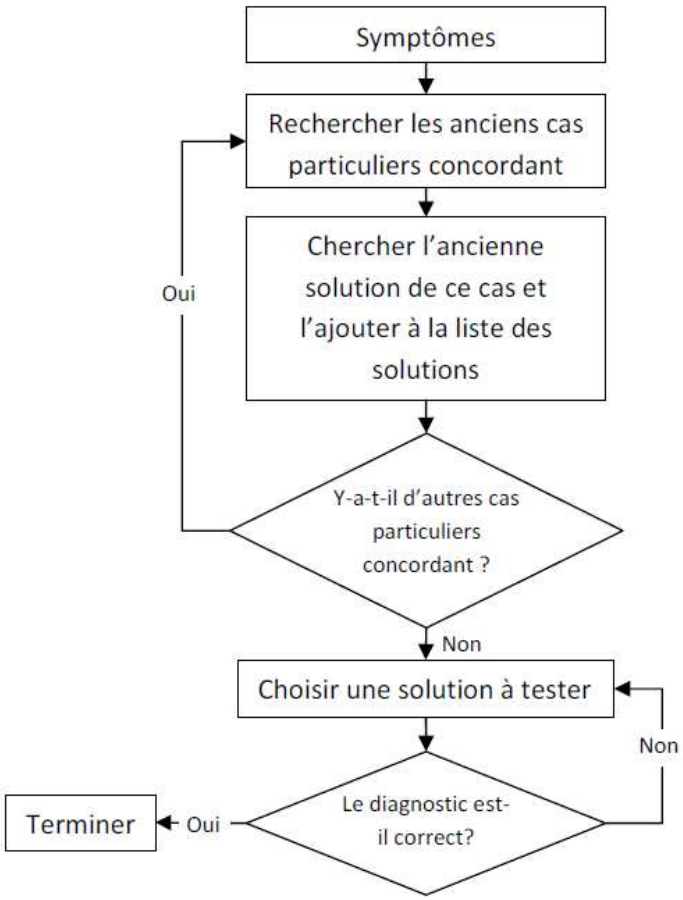


FIG. 1.2 – Diagnostic à base de reconnaissance de cas particulier

L'avantage de cette approche est qu'elle est particulièrement adaptée aux domaines d'application où les problèmes ne peuvent être facilement décomposés. Les solutions sont réutilisables. Quand un nouveau problème apparaît, il est probable qu'il puisse être résolu en utilisant

une solution ancienne d'un cas ancien similaire. En revanche, si les problèmes sont différents, alors il y a peu à gagner en essayant de réutiliser les solutions du passé.

L'inconvénient de cette approche est qu'elle dépend beaucoup de l'expérience de l'expert. L'historique, plus que la connaissance approfondie sur le modèle du système, est la source principale de connaissances. Afin de diagnostiquer efficacement, il est nécessaire d'avoir étudié de nombreux exemples historiques sur le système. Cette exigence paraît irréaliste dans le contexte de problème de diagnostic juste à temps.

1.2.3 Méthode de diagnostic à base de modèle

Le diagnostic à base de modèles, dite aussi analyse diagnostique formelle, s'appuie sur une description formelle du système à diagnostiquer afin de prédire le comportement du système. Nous allons utiliser un modèle pour définir les comportements du système en écrivant les modes de comportements de chaque ressource et les interactions entre eux au niveau du système. Nous appelons *observation*, un ensemble de données, issues en général de capteurs. Ces observations sont reliées à des variables du modèle. Un conflit est détecté lorsque des variables calculées par le modèle, à partir d'une observation, different de celles observées sur le système. Autrement dit, un conflit apparaît lorsqu'une observation et le modèle sont inconsistants.

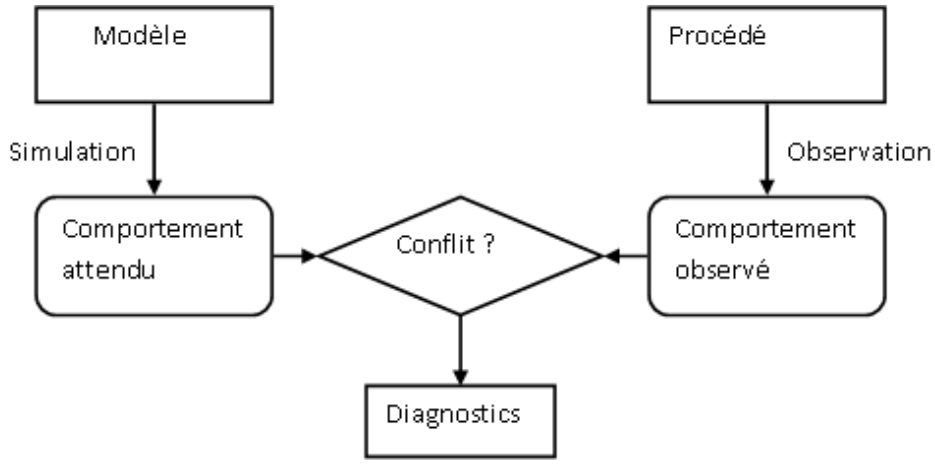


FIG. 1.3 – Diagnostic à base de reconnaissance de cas particulier

Approche FDI (Fault Detection and Isolation)

Dans la communauté FDI (Patton *et al.* [1989]; Gertler [1993]; Adjallah [1993]; Cassar *et al.* [1995]; Frank [1996]; Isermann [2004]), un problème de diagnostic se divise en deux parties : la partie détection et la partie localisation des défauts. Chaque composant du système est décrit par un ensemble de contraintes, qui exprime son comportement par l'intermédiaire des variables.

Au niveau du système, un ensemble des relations de redondance analytique (*RRA*), que nous appelons les *tests de détection* (ou simplement *test*), est établi permettant de générer les symptômes du système. En fait, une *RRA* est une relation qui ne contient que des variables observables dont les valeurs peuvent être calculées à partir de l'observation (Staroswiecki et Comtet-varga [2001]; Dustegor *et al.* [2004]; Ploix *et al.* [2005a]). Les variables non-mesurables sont éliminées. L'ensemble des composants qui appartiennent à une RRA_i est noté $sup(RRAi)$.

Le résultat de l'évaluation d'un *RRA* est appelé *résidus* (Frisk [2000]; Staroswiecki et Comtet-varga [2001]; Dustegor *et al.* [2004]; Nyberg et Frisk [2006]). En fonctionnement normal, les observations (OBS) satisfont les *RRAs*. Lors de la présence de défauts, il apparaît un conflit entre les comportements attendus et les comportements observés. En ce cas, *le résidu* est non nul, il explique un conflit. La signification de ce *résidu* est qu'il existe au moins un composant fautif parmi les composants intervenant dans la *RRA*.

Dans la deuxième étape de l'approche FDI, étape de localisation, il nous faut établir une table de signature des défauts (Frank [1996]). Etant donné un ensemble $R = \{RRA_1, RRA_2, \ldots, RRA_n\}$ de n RRAs et un ensemble $\mathscr{F} = \{F_1, F_2, \ldots, F_n\}$ de m défauts, la signature du défaut F_j est donnée par le vecteur colonne $FS_j = (s_{1j}, s_{2j}, \ldots, s_{nj})^T$ tel que :

$$s_{ij} = \begin{cases} 1 \textit{ si les composants impliqués dans } F_j \textit{ sont impliqué dans } RRA_i \\ 0 \textit{ sinon} \end{cases} \tag{1.2.1}$$

Une table de signature des défauts est formée par l'ensemble des vecteurs colonnes des défauts et chaque ligne correspond à un RRA. La signature d'une observation, notée par S_{obs}, est présentée aussi sous forme d'un vecteur colonne, fournie par la formule :

$$S_{obs} = \begin{pmatrix} 0 \textit{ si } RRA_1 \textit{ satisfait par observation ou } 1 \textit{ si non} \\ \cdots \\ \cdots \\ 0 \textit{ si } RRA_n \textit{ satisfait par observation ou } 1 \textit{ sinon} \end{pmatrix} \tag{1.2.2}$$

Dans l'approche de FDI, le défaut est localisé par comparaison entre les vecteurs colonnes associés aux symptômes abservés S_{obs} (ceci est appelé aussi signature effective) et les vecteurs colonnes de signature FS_j dans la table de signatures. Un ensemble des défauts F vérifiera :

$$\forall F_j \in \mathscr{F}, FS_j = S_{obs} \tag{1.2.3}$$

Cette approche donne un outil pour traiter les modèles quantitatifs. Ces modèles sont décrits par des équations mathématiques, souvent par des équations différentielles. Pourtant, ceci

est parfois un inconvénient de cette approche : elle exige des modèles mathématiques précis et complets. Ceci n'est pas facile à obtenir. Un autre inconvénient de cette approche est la garantie des diagnostics. Cela est dû à l'hypothèse d'exonération qui considère que si un résidu est quasiment nul alors il n'y a aucun défaut dans le sous-système testé, les composants impliqués dans ce sous-système sont exonérés. Cette hypothèse concerne l'hypothèse de non-compensation des défauts et de manifestation permanente des symptômes. Ces hypothèses peuvent conduire à de faux diagnostics car un test ne permet pas toujours de constater une anormalie en tout point de fonctionnement du sous-système testé. De plus, la comparaison entre les vecteurs colonnes d'observation et les vecteurs colonnes de signature dans la table de signature ne permet pas de détecter facilement les défauts multiples.

Approche DX

L'approche DX est une méthode de diagnostic à base de modèle. Il est aussi appelé diagnostic à partir des principes premiers. Cette méthode a été formulée au début des années quatre-vingts dans la communauté intelligence artificielle. Depuis son apparition, les travaux marquants sont (Reiter [1987]; De Kleer et Williams [1987]; Greiner *et al.* [1989]; De Kleer et Williams [1992]). L'idée principale est de comparer le fonctionnement réel du système par l'intermédiaire des observations avec le fonctionnement prédit grâce à un modèle de bon comportement. Les diagnostics sont alors calculés logiquement.

Dans cette approche, un problème de diagnostic est décrit par un triplet (SD, COMPS, OBS) (Reiter [1987]; De Kleer et Williams [1987]) avec SD (system description) un ensemble de formules logiques de premier ordre, COMPS (system components) un ensemble de composants du système (ensemble fini de constantes), OBS (observation) un ensemble d'observations du système décrit par un ensemble de formules de premier ordre. L'ensemble SD inclut un prédicat unitaire noté $AB(\dots)$ qui signifie anormal (ou abnormal en anglais). Si c est un composant qui appartient à l'ensemble des composants COMPS, $AB(c)$ signifie que le composant c se comporte anormalement. $SD \cup \{\neg AB(c) |\ c \in COMPS\}$ représente donc la description du comportement normal du système. Le système observé $(SD,\ COMPS,\ OBS)$ est détecté anormal si et seulement si $SD \cup \neg AB(c) |\ c \in COMPS \cup\ OBS$ est inconsistant. Puis un ensemble de conflits est généré ; il exprime l'inconsistance entre le modèle de comportement normal d'un ensemble de composants et les observations. L'étape de localisation repose sur l'analyse croisée des défauts des composants impliqués dans le support des conflits. Donc, un diagnostic doit expliquer tous les conflits. Les algorithmes pour calculer les diagnostics à partir des conflits sont présentés dans (De Kleer et Williams [1987]; Reiter [1987]). Des extensions sont proposées dans (De Kleer et Williams [1992]; Struss et Dressler [1989]; Desindes [2006]) pour intégrer le modèle de disfonctionnement (des défauts spécifiques). Cela permet d'exprimer les causes plus précises des conflits.

Comme le raisonnement diagnostique par l'approche DX, que l'on appelle aussi diagnostic

logique, ne repose pas sur l'hypothèse d'exonération, il apporte une garantie sur les résultats : si les modèles de comportement sont justes et si un dysfonctionnement se manifeste, alors nécessairement l'état réel du système fait partie de la liste des diagnostics calculés. Pourtant, le raisonnement diagnostique logique à partir des conflits peut conduire aux diagnostics physiquement impossibles. Ce problème est abordé et traité dans (Struss et Dressler [1989]; Desindes [2006]).

Approche bridge entre FDI et DX

Les premières études comparatives entre l'approche DX et FDI sont présentées dans (Cordier *et al.* [2000]). Puis, certaines contributions sont présentées qui consistent à rapprocher FDI et DX afin d'exploiter des avantages de chaque approche (Cordier *et al.* [2000]; Ploix *et al.* [2003]; Nyberg et Krysander [2003]; Cordier *et al.* [2004]).

L'approche DX qui s'appuie sur un raisonnement diagnostique logique et n'exige aucune hypothèse d'exonération permettra d'obtenir une garantie des résultats. Elle permet aussi de traiter les défauts simples et multiples. L'approche DX cherche directement des conflits à partir du modèle et des observations. L'étape de recherche des RRAs est implicite dans DX. Les RRAs donnés par l'approche DX peuvent être considérés comme des conflits potentiels (un RRA peut être satisfait ou non par les observations). La cohérence globale entre signatures de FDI permet d'adapter FDI aux environnements bruités. Généralement, l'approche bridge consiste à rapprocher FDI et DX afin d'exploiter les avantages de FDI sur l'aspect *détection* et des avantages de DX sur l'aspect *localisation*.

1.2.4 Positionnement de l'approche de diagnostic

Après une étude de différentes approches de diagnostic, l'approche à base de modèles, et plus précisément, l'approche bridge entre FDI et DX, a été préférée dans ce manuscrit. Une telle approche permet d'exploiter les avantages des approches FDI et DX. Généralement, cette approche ne dépend guère du type de modèle étudié. Elle s'adapte bien aux modèles qualitatifs ainsi qu'aux modèles quantitatifs. De plus, elle offre également des avantages sur l'évolutivité du système car elle est peu dépendante de l'expérience de l'expert. La mise à jour du modèle est faite facilement par l'ajout ou la suppression des modèles élémentaires. Ce point fort est exploité dans le contexte de *plug and play* présenté dans (Allahham *et al.* [2010])

1.3 Prise en compte de plusieurs niveaux d'abstraction dans l'analyse diagnostique

Pour modéliser un système, plusieurs types de formulation sont présentés dans la communauté de l'analyse diagnostique à base de modèle. Dans (Chittaro et Kumar [1998]; Chittaro et Ranon [2004]), quatre types de connaissances sont distingués :

- la connaissance structurelle est la connaissance sur la topologie du système. Elle décrit les composants constituant le système et les liens entre eux.
- la connaissance comportementale exprime les comportements potentiels des composants. Elle décrit comment les composants se comportent à l'aide des lois fondamentales (physique, chimique, ...).
- la connaissance fonctionnelle exprime le rôle des composants d'un système.
- la connaissance téléologique exprime l'objectif d'un système, d'un sous-système ou un composant. Ce type de raisonnement concerne les connaissances à haut niveau qui décide la conception et l'installation d'un système.

Dans la littérature scientifique, l'abstraction est présentée comme un outil qui permet de réduire le coût des calculs du diagnostic à base de modèle (Genesereth [1984]; Davis [1984]; Mozetic [1992]; Struss [1992]; Nayak [1994]; Autio [1995]; Autio et Reiter [1998]; Chittaro et Ranon [2004]). Le diagnostic commence souvent par le niveau d'abstraction le plus élevé du système, puis les solutions sont affinées hiérarchiquement. L'avantage est que, aux niveaux abstraits, le modèle du système est décrit de façon plus simple, et des solutions peuvent être calculées avec moins d'efforts. Ces solutions sont ensuite utilisées au plus bas niveau pour rechercher des solutions plus détaillées. Certains auteurs montrent que l'approche de diagnostic basé sur le modèle d'abstraction permet aussi d'éviter des diagnostics physiquement impossibles (Struss [1992]; Chittaro et Ranon [2004]). Ce problème n'est pas encore traité dans (Reiter [1987]; De Kleer et Williams [1987]).

Dans la littérature, l'abstraction du modèle est construite en s'appuyant sur les quatre types de connaissances citées ci-dessus. Pourtant, ces types de connaissances ne sont pas toujours distingués clairement par les auteurs. Plus souvent, on trouve d'un côté certains travaux qui se concentrent sur la connaissance structurelle et comportementale, et d'un autre côté, des travaux sur la connaissance fonctionnelle et téléologique. Dans la suite, nous souhaitons présenter certains travaux qui existent en distinguant ces deux catégories.

1.3.1 Construction du modèle hiérarchique en s'appuyant sur la connaissance structurelle et comportementale

Dans cette catégorie, les travaux proposés sont de deux types : l'abstraction structurelle et l'abstraction comportementale. Au point de vue topologique, l'abstraction structurelle ex-

prime une décomposition hiérarchique sur la structure du système. Par exemple, une voiture peut être décomposée structurellement en sous-systèmes : système électrique, système d'allumage, système d'injection, etc. Le système électrique peut être décomposé encore en batterie, fil électrique, bougie d'allumage, démarreur, et ainsi de suite. En parallèle à la décomposition structurelle, le modèle comportemental décrit le comportement du système et des sous-systèmes. L'abstraction comportementale consiste à décrire un système à différents niveaux de détail du comportement. Par exemple, un modèle qualitatif peut être décrit plus précisément par le modèle quantitatif. Nous allons regarder dans cette partie certains modèles existant dans la littérature sur l'abstraction structurelle et l'abstraction comportementale.

Modèle hiérarchique pour l'analyse diagnostique de Mozetic (Mozetic [1992])

Dans (Mozetic [1992]), Mozetic donne une théorie générale sur la description hiérarchique des systèmes qui comporte trois principes de raffinement/abstraction. Un algorithme de diagnostic est proposé pour ces modèles.

La représentation d'un système est donnée par un modèle m qui associe chaque mode x d'un système, ses observations correspondantes (entrées/sorties) y. Il est noté par :

$$m : x \rightarrow y \tag{1.3.1}$$

Les opérateurs de *raffinement / abstraction* permettent de partir d'un modèle m représentant le système et d'obtenir un modèle plus abstrait m' représentant le même système mais moins détaillé. Afin de relier les deux représentations, on a besoin d'une fonction h (inversible ou non-inversible) qui exprime le lien d'abstraction entre m et m'. En détail, $h(x,x')$ fait le lien entre x et x', et $h(y,y')$ fait le lien entre y et y'. Le modèle hiérarchique avec deux niveaux adjacents est présenté sur la figure 1.4.

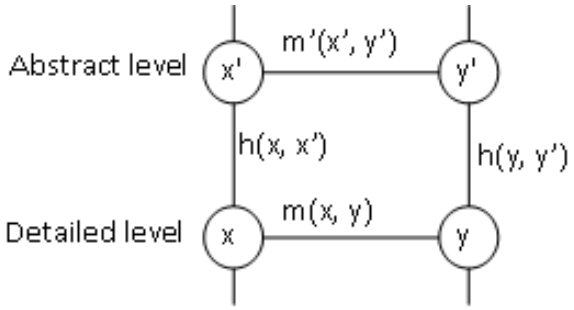


FIG. 1.4 – Représentation hiérarchique d'un modèle

Ensuite, trois principes d'abstraction/raffinement sont introduits. Ils expliquent comment un modèle hiérarchique est construit. Ce sont :

1. Introduction / Suppression d'une variable : Certaines variables peuvent être agrégées et disparaître au niveau supérieur ou de nouvelles variables sont introduites au niveau inférieur pour décrire plus en détail le modèle d'un système.
2. Raffinement / Abstraction d'une valeur : un ensemble de valeurs d'une variable peut être abstrait en une seule valeur. Par exemple, un intervalle de valeurs quantitatives est abstrait à une valeur qualificative.
3. Elaboration / Simplification (expansion / agrégation de la structure du modèle) : En terme de système, un modèle au niveau abstrait $m_1(x_1, y_1)$ peut être défini par les relations plus simples (relation entre les modes et les variables) que le modèle détaillé $m_2(x_2, y_2)$. Quand le système est constitué d'un ensemble de composants $c_1, c_2, \ldots, c_n$, la composition du système peut être définie par :

$$m(x,\ y) \leftarrow c_1(x_1, y_1), \ldots, c_n(x_n, y_n) \tag{1.3.2}$$

où $c_1, \ldots, c_n$ définissent des relations (appelée *mapping* par Mozetic) entre les modes de chaque composant et leurs observations (entrée/sortie). L'agrégation de certains composants (par exemple $c_1, \ldots, c_n$) peut conduire à une représentation plus simple :

$$m(x,y) \leftarrow c_1(x_c, y_c), c_{k+1}(x_{k+1}, y_{k+1}), \ldots, c_n(x_n, y_n) \tag{1.3.3}$$

Pour assurer la cohérence du raisonnement, il doit être garanti que les diagnostics qui sont impossibles à un niveau abstrait sont aussi impossibles à un niveau plus détaillé. Ceci est appelé la condition de consistance, qui doit être satisfaite entre les différents niveaux (voir détails dans (Mozetic [1992])).

Pour conclure, Mozetic a donné dans son travail la formulation d'un modèle hiérarchique avec trois principes pour le raffinement et l'abstraction du modèle. Ces principes donnent une vue générale sur le raffinement et l'abstraction et ils ne distinguent pas en détail les deux aspects structurel et comportemental. Nous pouvons dire que les deux premiers principes, et une partie du troisième principe visent à l'abstraction comportementale. L'agrégation des composants évoquée dans le troisième principe vise à l'abstraction structurelle du système.

Modèle hiérarchique proposé par Chittaro (Chittaro et Ranon [2004])

Dans (Chittaro et Ranon [2004]), Chittaro a proposé une formalisation de l'abstraction structurelle pour l'analyse diagnostique. Elle est appelée MIS (Model Increasing Structural). Supposons que la procédure d'abstraction commence par un problème de diagnostic $D_0 = (SD_0, COMPS_0, OBS_0)$, et que nous voulions le caractériser structurellement par un problème de diagnostic plus abstrait $D_1 = (SD_1, COMPS_1, OBS_1)$. Une fonction d'interprétation (interpretation mapping) pour l'abstraction structurelle est définie par $\pi_A : Models(D_0) \rightarrow Interpretations(L_1)$ qui permet de décrire D_1 dans le langage L_1. D_1 est le résultat de l'agrégation d'un ensemble des composants $AGGR \subseteq COMPS$ en un seul super-composant. $COMPS_1$

contient les nouveaux composants qui représentent l'agrégation de certains composants de $COMPS_0$, et les composantes de $COMPS_0$ qui ne sont pas impliqués dans les agrégations.

En terme de comportement des composants, le comportement d'un super-composant dépend du comportement de ses sous-composants. Le super-composant se comporte d'une certaine manière, ses sous-composants doivent être dans un certain comportement correspondant. Il est exprimé par la combinaison des modes de comportement des sous-composants. Chaque combinaison est une conjonction logique des modes, par exemple $m_1(c_1) \wedge \cdots \wedge m_k(c_k)$ avec m_i, un mode de comportement de c_i. Chaque conjonction de modes de ce type est appelée aussi *un candidat partiel* de $\{c_1, \ldots, c_k\}$. L'ensemble de *candidats partiels* dépend de l'ensemble des composants impliqués dans *AGGR* et de l'ensemble de modes de comportement de chaque composant $c_i \in AGGR$. Il est noté par

$$CANDS(SD_0, AGGR)$$

Nous allons créer des ensembles de *candidats partiels* $BM_1, \ldots, BM_n$ tel que $BM_i \subseteq CANDS(SD_0, AGGR)$, $\bigcup_{i=1\ldots n} BM_i = CANDS(SD_0, AGGR)$. Chaque ensemble de *candidats partiels* correspond à un mode de comportement spécifique du super composant.

Considérons l'exemple de deux tuyaux p_1 et p_2. Supposons que chaque tuyau comporte deux modes de comportement $\{ok, fuite\}$. L'ensemble des candidats partiels possibles de l'agréation $AGGR = \{p_1, p_2\}$ sont

$$\{ok(p_1) \wedge ok(p_2), ok(p_1) \wedge fuite(p_2), fuite(p_1) \wedge ok(p_2), fuite(p_1) \wedge fuite(p_2)\}.$$

Nous pouvons regrouper ces *candidats partiels* en deux ensembles BM_1 et BM_2 de la manière suivante :

$$BM_1 = \{\{ok(p_1) \wedge ok(p_2)\}\}$$
$$BM_2 = \{\{ok(p_1) \wedge fuite(p_2)\}, \{fuite(p_1) \wedge ok(p_2)\}, \{fuite(p_1) \wedge fuite(p_2)\}\}$$

Dans cet exemple, le super composant peut avoir deux modes de comportement : le premier mode représente le fait que les deux sous-composants sont en mode de fonctionnement normal, le deuxième mode représente le fait qu'il y a une fuite au niveau des sous-composants.

Du point de vue structurel, la notion de *port* est introduite dans (Chittaro et Ranon [2004]). Structurellement, la connexion entre les composants dans le système est réalisée par *les ports*. Un port correspond à une variable dans la description comportementale est lié à une observation possible. En comparaison avec D_0, certains ports seront retirés au niveau d'abstraction D_1. OBS_1 est un sous-ensemble de OBS_0, OBS_1 représente les ports qui ne sont pas retirés dans une abstraction de D_0 à D_1.

Chittaro a proposé dans son travail une formalisation de l'abstraction structurelle MIS. Une hiérarchie de MIS est un ensemble ordonné de m problèmes de diagnostic $\{SD_i, OBS_i, COMPS_i\}$ avec $i = 0, \ldots, n-1$. $\{SD_0, OBS_0, COMPS_0\}$ est le moins abstrait et $\{SD_{n-1}, OBS_{n-1}, COMPS_{n-1}\}$ est le plus abstrait. En comparaison avec (Mozetic [1992]; Struss [1992]; Autio [1995]; Autio et Reiter [1998]), une extension au modèle avec multi-modes de comportement est introduit. La description aux différents niveaux de comportement n'est pas clairement abordée dans (Chittaro et Ranon [2004]). Les contraintes sont introduites pour décrire les modes des composants à chaque niveau d'abstraction mais à la condition que ceci ne change pas le domaine de valeurs des variables.

1.3.2 Construction du modèle hiérarchique en s'appuyant sur la connaissance fonctionnelle et téléologique

Pour certains auteurs, la notion de fonction est fortement liée à la notion de téléologie : une fonction d'un système est identifiée par une tâche ou un objectif que le système doit accomplir (De Kleer [1984]; Steels [1989]; Downing [1990]; Franke [1991]; Keuneke [1991]; Sticklen et Bond [1991]; IEEE [1998]; Flaus [2008]). Certains autres auteurs distinguent plus précisément la fonction de la téléologie en considérant le modèle fonctionnel comme une relation intermédiaire entre le modèle comportemental d'un système et le modèle téléologique. Pourtant, dans ces approches, la présentation et l'utilisation de la connaissance fonctionnelle et téléologique est discutée de façon générale. Une formalisation plus concrète a été récemment présentée dans (Chittaro et Ranon [2004]). Nous allons voir ensuite dans cette partie la formulation de la connaissance fonctionnelle et téléologique selon ces deux points de vue.

Pour le premier point de vue, une fonction du système est définie comme une relation entre le comportement du système (ou sous-système) et l'objectif à obtenir conçu par le concepteur. Pourtant, ces approches sont souvent limitées par un langage spécifique qui dépend du type de système étudié. Certaines de ces approches sont appelées approches fonctionnelles à base de flux ou FBF (flow-based functional approche). Il s'agit de la présentation fonctionnelle des flux de matière (la chaleur, les électrons, les liquides, etc.) qui coulent à travers la structure du système. Dans l'approche FBF, les fonctions sont présentées par un ensemble de primitives (ou primitive fonctionnelle). Chaque primitive fonctionnelle représente une action effectuée sur le flux qui coule à travers la structure du système. Par exemple, du système hydrologique, quatre primitives fonctionnelles peuvent être définies : générer le flux, conduire le flux, stocker le flux, et empêcher le flux. Puis, certaines règles sont conçues afin de réaliser une abstraction automatique sur le modèle fonctionnel : abstraction de deux primitives fonctionnelle en série, abstraction de deux primitives fonctionnelles en parallèle, conversion étoiles-mailles, etc. Par exemple, l'abstraction de deux primitives en série *conduire le flux* et *empêcher le flux* emmènera à une autre primitive fonctionnelle *empêcher le flux*.

En résumé, la formalisation des primitives fonctionnelles permet de faire le lien entre le

comportement des composants du système à son objectif. En s'appuyant sur ces règles, l'abstraction peut être réalisée automatiquement. Un inconvénient de la notion de cette formulation est qu'elle dépend largement de la nature de système étudié.

Pour le deuxième point de vue, la notion de fonction et la notion de téléologie ne sont pas distinguées. Une fonction est définie comme un processus présentatif du rôle d'un système ou d'un ensemble de ressources. Les ressources sont peut-être des ressources matérielles, humaines, informationnelles, etc. La notion de fonction proposée exprime le lien entre les fonctions et les ressources assistant à la réalisation de la fonction. Elle est adaptée à une grande variété de systèmes et pour certains objectifs d'utilisation comme l'analyse fonctionnelle, l'analyse des risques, l'analyse diagnostique, etc. Deux modèles peuvent être cités de la littérature : IDEF0 (Intergration Definition For Function Modeling) (IEEE [1998]) et FIS (Fonction-Interaction-Structure) Flaus [2008]).

IDEF0 est souvent exploitée pour la modélisation des processus de production et de service (Zakarian et Kusiak [2001]; Kim et Jang [2002]). Cette technique convient pour la modélisation des activités et du flux des informations entre les activités. Pour modéliser un système, elle s'appuie sur trois types de diagrammes : graphique, texte et le glossaire. Les schémas graphiques définissent les fonctions et les relations fonctionnelles via les boîtes et des flèches. Le texte et le glossaire fournissent des informations supplémentaires aux schémas graphiques. Un schéma graphique d'une fonction peut être représenté sur la figure (1.5).

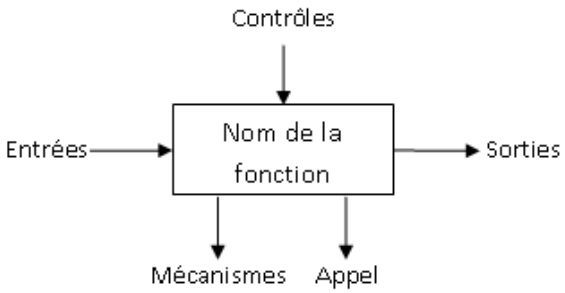


FIG. 1.5 – Diagramme d'une fonction dans IDEF0

Sur ce diagramme, la boîte rectangulaire indique la fonction, étiquetée par un nom. Les flèches qui viennent de gauche sont des entrées. Les entrées sont transformées ou consommées par la fonction. Les flèches qui partent vers la droite sont des sorties. Ce sont des données ou des produits fournis par la fonction. Les flèches qui viennent du haut sont des contrôles. Ils précisent les conditions nécessaires pour que la fonction puisse produire des sorties correctement. Les flèches qui viennent du bas sont des mécanismes. Elles identifient des moyens utilisés pour l'exécution de la fonction. Les flèches qui pointent vers le bas sont appelées des appels. La boîte appelée va fournir des détails à la boîte qui appelle. Le modèle IDEF0 permet de décrire un système de façon hiérarchique avec des diagrammes aux différents niveau de détail (figure 1.6).

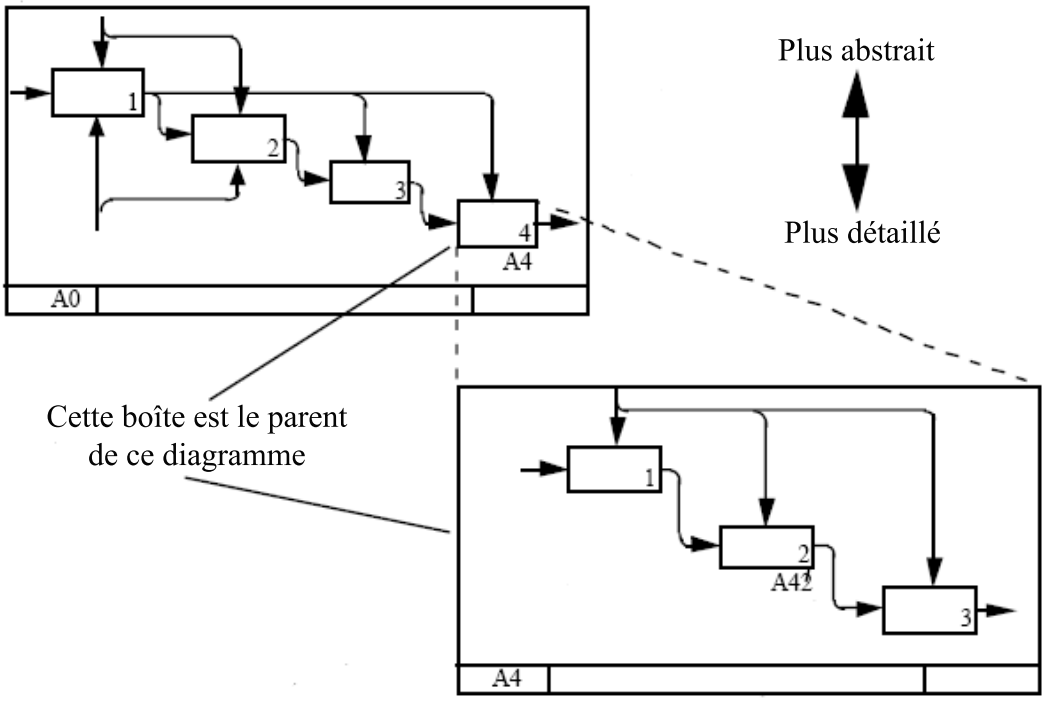


FIG. 1.6 – Diagramme d'une fonction

Un inconvénient de IDEF0 est qu'il ne permet pas de prendre en compte le temps ni de représenter les relations logiques (relations ET / OU). Pour cette raison, il ne donne pas une vue générale sur le déroulement ou la séquence des fonctions. Une amélioration a été proposée dans les versions IDEF1, IDEF2.

Dans (Flaus [2008])), FIS (Fonction-Interaction-Structure) est présenté comme un outil de modélisation efficace pour l'analyse des risques et un moyen pour intégrer des outils de l'analyse diagnostique (Giap *et al.* [2010c]). La notion de fonction présentée dans FIS est similaire à celle proposée IDEF0. Cependant, les liens entre les aspects structurels et fonctionnels sont explicités. Les entrées des fonctions sont constituées des ressources utilisées et les sorties des ressources produites ou modifiées. Par conséquent, la représentation de l'ensemble d'un système est un graphe biparti Fonctions/Ressources (figure 1.7).

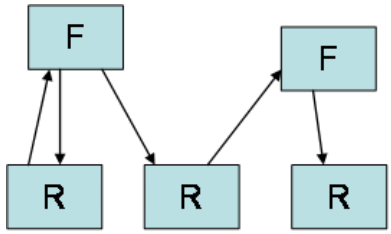


FIG. 1.7 – Relations Fonctions/Ressources dans FIS

Plusieurs fonctions peuvent être regroupées au sein d'un système de façon à rendre l'analyse modulaire.

Des fonctions et/ou ressources peuvent être partagées entre les systèmes et la séquence d'exécution des fonctions est spécifiée par un diagramme séquentiel indépendant du graphe Fonction/Ressources (figure 1.8). Enfin la partie dysfonctionnelle du modèle permet de représenter les relations ET et OU entre les fonctions. L'ensemble de l'approche est mise en oeuvre dans le cadre du logiciel XRisk.

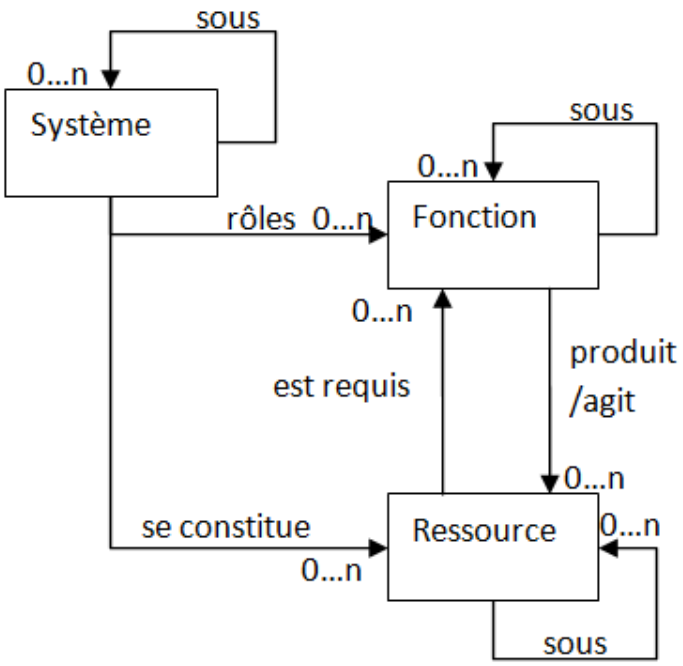


FIG. 1.8 – Présentation UML du modèle FIS

1.3.3 Positionnement de la méthode de modélisation

Pour construire le modèle hiérarchique, nous pouvons distinguer dans la littérature deux approches : les approches de bas en haut (Bottom-Up) et les approches de haut en bas (Top-Down approachs). Les approches de bas en haut consistent à construire le modèle hiérarchique du système à partir du niveau détaillé des composants (ou des ressources) pour conduire à un niveau plus abstrait. Le but principal de l'exploitation d'un tel modèle hiérarchique est souvent de réduire le coût et le temps de calcul dans l'analyse diagnostique. Les approches de haut en bas consistent à décomposer et à décrire les systèmes de façon hiérarchique de haut en bas. La décomposition commence souvent par les fonctions, la structure globale du système et conduit à un niveau plus détaillé. La décomposition hiérarchique d'un système permet d'analyser (analyse fonctionnelle, analyse des risques, etc.) en évitant la complexité du système. Dans le contexte du diagnostic, les approches de haut en bas pour décrire un système

à diagnostiquer paraissent plus convenables car elles n'exigent pas le modèle complet, détaillé du système comme préalable.

Comme le système est construit au fur et à mesure dans la procédure de diagnostic, nous souhaitons reformuler et présenter dans notre travail une approche de modélisation dite orientée objet. Dans ce but, le phénomène d'abstraction sera défini en s'appuyant sur les éléments de base qui constituent le système. Comme l'abstraction sur les variables, l'abstraction sur les items (un item est soit une fonction soit une ressource), etc. Par ces définitions, les abstractions structurelle et comportementale sont implicitement intégrées. Cette reformulation permettra de faire le pont entre l'approche bridge de diagnostic (FDI/DX) basée sur les contraintes que nous avons évoquées précédemment et les méthodes basées sur l'abstraction. De plus, elle donnera une vue générale sur un langage de description qui permet à l'expert de construire un modèle hiérarchique au fur et à mesure en exprimant ses connaissances sur le système.

1.4 Interaction dans l'analyse diagnostique

1.4.1 Interaction home-automate dans l'analyse diagnostique

L'interaction homme-automate est présentée dans l'analyse diagnostique comme une façon d'exploiter au fur et à mesure la connaissance de l'expert dans le processus de diagnostic. Nous allons revoir certains travaux existant.

Les travaux de (Ploix et Chazot [2006]) présentent une approche de diagnostic itératif qui permet de réaliser une modélisation structurelle au fur et à mesure dans le processus de diagnostic. La décomposition du système dépend des résultats des tests à chaque itération. Cette approche permet d'exploiter au maximum l'expertise implicite et d'éviter les formalisations inutiles, il s'agit de la partie de la modélisation qui n'est pas exploitée dans l'analyse diagnostique. En détail, le niveau structurel est exploité dans (Ploix et Chazot [2006]) parce qu'il est plus facile pour un expert de modéliser un système. Il décrit directement les composants impliqués dans le système et les connexions entre eux. Un processus de description itérative peut être mis en évidence, il commence par une description au niveau des macro-composants et selon les résultats du diagnostic, les parties où les défauts sont localisés sont décrites en détail. Pour expliciter au fur et à mesure l'expertise, on suppose que implicitement, un expert soit capable de :

- effectuer des tests de détection pour améliorer la localisation des défauts lorsque les diagnostics sont proposés par l'automate à une itération.
- formaliser la structure des parties du système qui sont testés. Il s'agit des composants et les connexions entre eux via des variables.

A un moment donné, Il est difficile pour un expert de résumer tous les résultats des tests

de détection et de les analyser de manière systématique afin d'en déduire toutes les fautes possibles. Un outil d'aide au diagnostic qui s'appelle SMARTlab a été présenté dans (Ploix et Chazot [2006]). SMARTlab permet de calculer tous les diagnostics minimaux à partir des symptômes et lister les diagnostics en un ordre de priorités. Dans SMARTlab, les tests négatifs sont utilisés pour calculer les diagnostics selon l'algorithme de Reiter (Reiter [1987]), qui formalise un raisonnement de diagnostic logique et permet de prendre en compte des défauts multiples. L'ordre des diagnostics est classé par la mesure contextuelle et la probabilité des modes. Ceci est présenté dans (Touaf et Ploix [2004a]) et est rappelé dans l'annexe A. Un exemple sur l'exploitation de SMARTlab est présenté dans (Ploix et Chazot [2006]).

Dans (Kleer et Williams [1987]; De Kleer et Williams [1992]), l'interaction homme-automate est présentée au niveau des mesures. Pour superviser un système industriel, certains capteurs peuvent être disponibles pour détecter les défauts. Pour localiser plus précisément, il faut avoir des mesures supplémentaires. Le problème proposé par de Kleer est de localiser un défaut avec un nombre de mesures minimal. La méthode proposée s'appuie sur la probabilité des défauts et la théorie de l'Entropie pour proposer les prochaines mesures au testeur.

Dans (Chittaro et Ranon [2004]), la procédure de diagnostic est quasiment automatique à partir du modèle du système et des observations. La suggestion des mesures n'est pas abordée dans son travail. L'interaction entre l'expert et l'automate a lieu au niveau du modèle téléologique afin de déterminer la partie active ou inactive du système. Ceci permet de déterminer le contour du système à tester.

1.4.2 Positionnement de l'objectif d'interaction homme-automate dans l'analyse diagnostique

Nous voyons que la plupart des travaux existant dans le domaine diagnostique sont présentés toujours avec l'hypothèse que le modèle du système est décrit complètement au préalable. Une autre piste de recherche qui est quasi nouvelle est présentée dans (Ploix et Chazot [2006]). Dans ce travail nous souhaitons suivre cette piste afin de résoudre des difficultés liées à la complexité des modèles. Nous considérons l'interaction homme-automate comme la seule façon permettant d'exploiter l'expertise qui ne peut pas être exprimée sous forme de modèle formel. Nous allons discuter ensuite les types d'interaction homme-automate dans l'analyse diagnostique en faisant le lien avec le modèle et l'approche de diagnostic que nous avons discuté précédemment. Au contraire de certaines approches présentées dans la littérature où les étapes de génération de symptômes et de diagnostic sont confondues, dans les problèmes que nous présentons, les trois étapes : modélisation du système, génération des symptômes et analyse diagnostique seront bien distinguées. Ensuite, nous étudierons l'interaction homme-automate suivant trois directions étudiées dans les chapitres 3,4 et5. (figure 1.9).

Dans le premier problème de *diagnostic itératif basé sur le modèle structuro-fonctionnel*

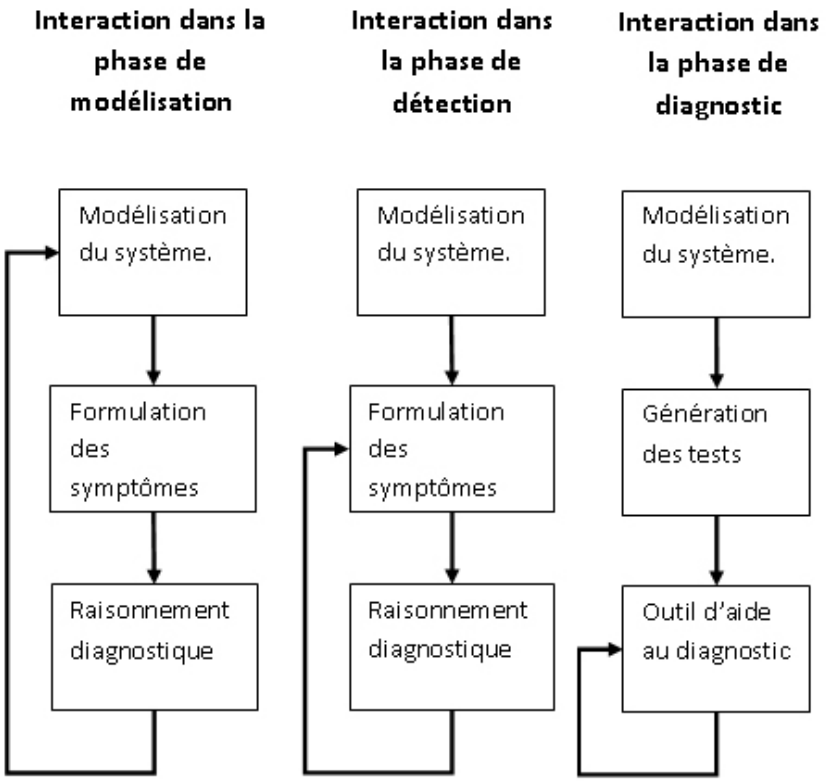


FIG. 1.9 – Trois types d'interaction homme-automate dans les problèmes de diagnostic itératifs

FIS, l'interaction homme-automate se manifeste durant la tâche de modélisation et de détection. L'opérateur décompose et teste le système selon différents niveaux d'abstraction, et puis il formalise les symptômes. Les diagnostics seront calculés à partir des symptômes obtenus. Cette approche est proposée pour les problèmes de diagnostic où le modèle de comportement du système peut être implicite. La réalisation des tests et la formulation des symptômes reposent sur la connaissance implicite de l'expert. Un problème de diagnostic hors-ligne d'une voiture a été présenté dans (Giap *et al.* [2010b]).

Dans le deuxième problème de *diagnostic itératif basé sur le modèle comportemental*, l'interaction homme-automate se manifeste durant la tâche de modélisation. Ce problème concerne plus particulièrement les systèmes complexes, où le modèle du système est décrit partiellement et certaines parties seront affinées au fur et à mesure. En effet, l'analyse diagnostique pour un modèle complet détaillé n'est souvent pas réaliste parce qu'elle est trop longue et trop couteuse à établir. De plus, il n'est pas nécessaire d'aborder des modèles qui ne sont pas utilisés dans l'analyse diagnostique. Nous pouvons constater que les symptômes sont formulés par l'expert dans le premier problème tandis qu'on a besoin de l'automate dans la tâche de détection de symptômes dans ce deuxième problème. Pour cette raison, en plus des outils du premier problème, les nouveaux outils seront conçus dans ce deuxième problème pour détecter automatiquement des symptômes à partir d'un modèle décrit à différents niveaux de

comportement.

Dans le dernier problème de *diagnostic itératif basé sur la connaissance experte implicite*, l'interaction homme-automate se manifeste durant la tâche de diagnostic. Dans ce problème, le modèle du système est construit au début mais il n'est pas complet. A l'aide des observations fournies, un ensemble de diagnostics peut être calculé mais cet ensemble est parfois trop important pour être appréhendé par un expert. Par exemple dans un problème de diagnostic pour un réseau de pluviomètres, qui est présenté dans le projet Hydrodiag, 16 diagnostics différents sont trouvés avec 5 ou 6 défauts simultanément. L'intervention de l'opérateur dans l'étape de diagnostic est alors nécessaire. Les connaissances implicites (non-formalisé) de l'expert seront exploitées dans l'analyse des données collectées pour isoler des défauts.

Les deux premiers problèmes consistent à résoudre les difficultés liées à la complexité (la structure et le comportement). Le dernier consiste à résoudre les difficultés liées à l'exploitation des types de connaissance non-formulée par l'expert.

1.5 Conclusions

Dans cette partie, nous avons réalisé une étude de bibliographie en positionnant les problématiques abordées dans ce mémoire. Ce positionnement repose sur le choix de l'approche de diagnostic, sur le choix de l'approche de modélisation hiérarchique, et sur la façon dont nous exploitons l'interaction homme-automate dans l'analyse diagnostique afin d'obtenir notre objectif. On peut résumer en trois points suivants :

- Pour une approche générale de diagnostic, l'approche bridge entre FDI et DX est choisie dans ce manuscrit. Ceci permet d'exploiter les avantages des approches FDI et DX, de résoudre le problème de diagnostic à base de contraintes, et de prendre en compte des défauts multiples.
- Pour la description du système, nous souhaitons une approche de modélisation qui permette de construire un modèle hiérarchique de haut en bas. Cette approche est dite orientée objet car elle s'appuie sur la définition de l'abstraction des éléments constituant le système (variable, contrainte, item, etc.). Ceci constitue une base pour concevoir un outil qui permette à l'expert de construire un modèle hiérarchique au fur et à mesure en exprimant ses connaissances sur le système.
- Pour l'aspect d'interaction homme-automate dans l'analyse diagnostique, nous souhaitons l'exploiter pour résoudre les difficultés liées à la complexité du modèle et à l'exploitation de la connaissance non formulée par l'expert.

Chapitre 2

Eléments de formulation

2.1 Eléments de modélisation pour un problème de diagnostic

Nous allons rappeler dans cette partie les éléments de modélisation existants dans la problématique de diagnostic à base de modèle de consistance (approche hybrique FDI et DX). Nous développerons ces éléments de façon à pouvoir les exploiter plus tard dans les problèmes de diagnostic interactif étudiés dans ce mémoire.

2.1.1 Variables

Dans un système, la réalité physique est constituée de phénomènes observables, modélisés par des grandeurs physiques $\mathscr{V}$ comme par exemple : l'intensité du courant électrique, la température de l'eau, etc. Chaque grandeur physique peut être caractérisée par des variables qui s'associent à des domaines de valeurs différents. Pour décrire au fur et à mesure un système à différents niveaux de détail, nous devons utiliser plusieurs variables qui caractérisent une même grandeur physique. Pour la définition, nous avons :

Définition 2.1.1. *(variable) : Chaque grandeur physique $\mathscr{V}$ peut être caractérisée par une ou plusieurs variables v définies par des domaines de valeurs différents $dom(v)$. Nous pouvons aussi noter une variable avec son domaine de valeur correspondant par $v_{\angle dom}$.*

Le domaine de valeur $dom(v)$ peut être continu, discret, ou sous forme d'intervalles.(Figure 2.1)

Si $\mathscr{V}$ est une grandeur physique représentée par $v_{\angle dom}$, on peut noter $\mathscr{V} = \varphi(v_{\angle dom})$ avec φ une fonction qui retourne la grandeur physique correspondant à la variable v.

Exemple 1. $u_{\angle\mathbb{R}} = Ri_{\angle\mathbb{R}}$, ou plus simplement $u_1 = Ri_1$, avec $u_1 = u_{\angle\mathbb{R}}$ et $i_1 = i_{\angle\mathbb{R}}$.

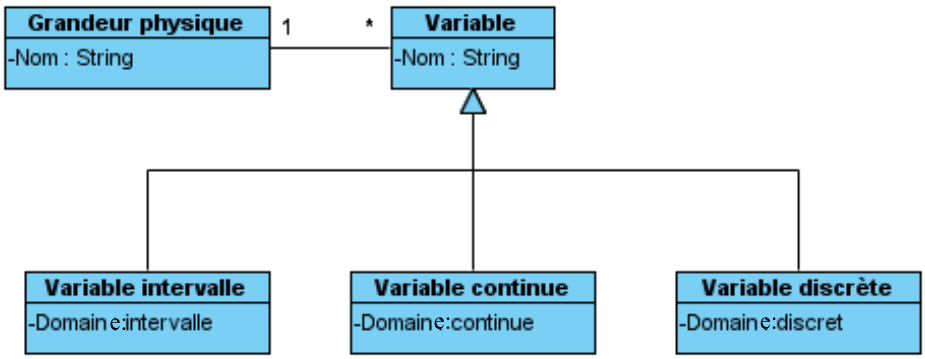


FIG. 2.1 – Les types de variable d'une grandeur physique

Pour faire le lien avec la notion de *port* proposée dans (Chittaro et Ranon [2004]) pour les composants, nous allons distinguer ici les deux notions *variable interne* et *variable externe*.

- *Une variable interne* est une variable qui est interne au modèle d'un composant, c'est-à-dire qu'elle n'est partagée avec aucun autre composant.
- *Une variable externe* est une variable qui est qualifiée d'externe au modèle d'un composant parce qu'elle est partagée par plusieurs composants. Ces variables représentent des échanges entre des composants ou entre un composant et l'environnement extérieur.

Les *variables externes* sont présentées comme des *ports d'entrée/sortie* (Chittaro et Ranon [2004]) et elles sont "vues" de l'extérieur.

Nous avons rappelé et distingué ici *grandeur physique* et *variable*. Les différentes variables d'une même grandeur physique peuvent être distinguées par leurs domaines de valeur. Cette distinction permet de formuler un modèle d'abstraction qui se rapporte aux variables.

2.1.2 Contrainte

Une contrainte est une relation entre des variables exprimées par des relations qualitatives ou quantitatives ou encore sous forme de tuples (Desindes [2006]). Les comportements d'un item sont décrits par *des contraintes de comportement* qui peuvent être associées à *des contraintes de validité*. *Les contraintes de validité* indiquent les conditions pour lesquelles *des contraintes de comportement* ont du sens.

Définition 2.1.2. *(Contrainte de comportement) Une contrainte k est une relation entre des variables* $\{v_1, ...v_n\}$ *qui caractérise un sous-ensemble* $\triangle$ *de* $dom(v_1) \times ... \times dom(v_n)$. *Une contrainte est satisfaite par un jeu de valeurs* $(\vartheta_1 \in dom(v_1), ..., \vartheta_n \in dom(v_n))$ *si* $(\vartheta_1, ..., \vartheta_n) \in \triangle$. *On dira que* $(\vartheta_1, ..., \vartheta_n)$ *est consistant avec la contrainte.*

Le comportement d'un composant physique est décrit par une ou plusieurs contraintes basées sur les variables $\{v_1, ...v_n\}$.

Selon la nature des contraintes, deux types de contraintes sont distingués (Yassine [2008]) :

Définition 2.1.3. Une contrainte terminale k_{obs} *est une contrainte qui associe une variable avec une valeur.*

Par exemple $x_1 = \widetilde{x_1}$ est une contrainte terminale avec $\widetilde{x_1}$, une valeur de variable x_1 : $\widetilde{x_1} \in dom(x_1)$. Les contraintes terminales représentent les observations (capteur/actionneur) sur un système.

Définition 2.1.4. Une contrainte de procédé k_{pro} *modélise le comportement d'un système. Ces contraintes établissent des relations entre plusieurs variables.*

Par exemple $x_1 + x_2 = x_3$ est une contrainte de procédé avec $\{x_1, x_2, x_3\}$ des variables.

Exemple 2. Pour illustrer la notion de la contrainte de validité évoquée précédemment, nous prenons l'exemple du modèle de deux *bacs* dessiné sur la figure (2.2). Le modèle comportemental du système est donné par :

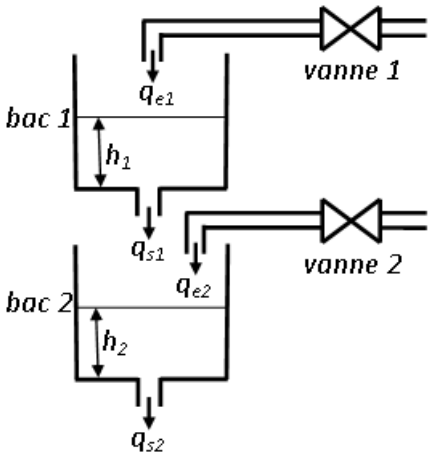


FIG. 2.2 – Système de deux bacs en cascade

$$
\begin{aligned}
&\left(ok(bac1), S\frac{dh_1}{dt} - q_{e_1} + kh_1 = 0, 0 \leq h_1 < 50\text{cm}\right)\\
&(ok(bac1), q_{s_1} - kh_1 = 0)\\
&\left(ok(bac2), S\frac{dh_2}{dt} - q_{e_2} - kh_1 + kh_2 = 0, 0 \leq h_2 < 50\text{cm}\right)\\
&(ok(bac2), q_{s_2} - kh_2 = 0)\\
&(ok(capteur1), obs(t, h_1) - h_1 = 0)\\
&(ok(capteur2), obs(t, h_2) - h_2 = 0)\\
&(ok(vanne1), obs(t, q_{e_1}) - q_{e_1} = 0)\\
&(ok(vanne2), obs(t, q_{e_2}) - q_{e_2} = 0)
\end{aligned}
\tag{2.1.1}
$$

Pour que la première et la troisième équation dans l'ensemble des équations (2.1.1) soient significatives, la hauteur d'eau dans le bac doit être inférieure à 50 cm. Deux contraintes de validité sont donc ajoutées au modèle du système : $0\ cm \leq h_1 \leq 50\ cm$, $0\ cm \leq h_2 \leq 50\ cm$.

Nous n'abordons pas les contraintes de validité dans les chapitres (3) et (4). Seul le chapitre 5 les examine.

Une contrainte est un élément de base dans la description du comportement du système. Elle est considérée comme un élément de référence pour détecter les défauts apparaissant dans un système physique.

2.1.3 Modes de comportements

Dans un problème de diagnostic à base de modèle, les modes de comportement sont introduits pour décrire le comportement d'un système. Le comportement d'un système (ou d'un composant) est spécifié via des contraintes de comportement qui caractérisent des ensembles de valeurs inclus dans les espaces des variables concernées. Ces ensembles seront nommés les *zones* (Figure 2.3). Parmi les zones dans l'espace des variables caractérisant un comportement, seules certaines parties sont physiquement possibles.

Le complément de la zone des possibles dans l'espace des valeurs des variables est appelé la zone impossible *ZI*, exprimé par le fait qu'il est physiquement impossible d'atteindre cette zone.

Un mode de comportement d'un composant désigne une famille significative de comportements. Le comportement d'un composant est défini à l'aide d'un ensemble de contraintes dans l'espace des valeurs des variables (Figure 2.3). Dans une zone possible, un composant

peut avoir un mode de fonctionnement/comportement correct et plusieurs modes de fonctionnements incorrects différents. Ils peuvent être classés en trois catégories :

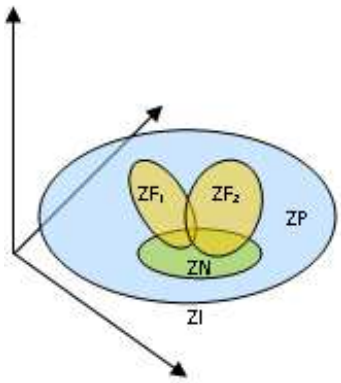


FIG. 2.3 – Les zones dans l'espace des variables

- Mode de fonctionnement/comportement normal, noté par *ok* (mode normal), caractérisé par le fait que les variables restent dans la zone de comportement normal *ZN*.
- Mode de défaut (ou de défaillance) spécifique, noté par *fm* (fault mode), caractérisé par le fait que les variables sont dans une zone de comportement anormal caractéristique d'un défaut ou d'une défaillance.
- Mode de défaut (ou de défaillance) complémentaire ou inconnu, noté par *cfm* (complementary fault mode), caractérisé par le fait que les variables sont hors zone spécifiée *ZN* et tombe dans la zone des possibles ZP mais sans pour autant appartenir à une zone *ZF*. Les modes *cfm* sont donc des modes anormaux auxquels ne correspondent aucune contrainte.

Par conséquent, l'ensemble des modes de comportement d'un composant *C* est écrit par :

$$Modes(I_i) = \{ok, [fm_1, \ldots, fm_n,]\ cfm\ \} \tag{2.1.2}$$

Pour faciliter la formulation de l'abstraction plus tard, nous considérons dans ce travail qu'un mode de comportement d'un composant est décrit par une contrainte de comportement et par une ou plusieurs contraintes de validité. Supposons que k soit une contrainte de comportement (ou l'ensemble des contraintes de comportement K) décrivant le mode $mode(I_i)$. En faisant le lien avec l'espace des variables, nous donnons ici des notions qui seront utilisées plus tard dans notre formulation.

- $var(k)$ indique l'ensemble de variables apparaissant dans la contrainte k. Par exemple, on peut avoir $var(k) = \left\{v_{1\angle\mathbb{R}}, \ldots, v_{n\angle\{0,1\}}\right\}$.
  Pour distinguer l'ensemble de variables de l'ensemble des grandeurs physiques apparaissant dans la contrainte k, on note $VAR(k)$, l'ensemble de grandeurs physiques.
- $valvar(k)$ indique un jeu de valeurs $\vartheta_1 \in dom(v_1), ..., \vartheta_n \in dom(v_n)$ associées à $var(k)$. Par exemple, $valvar(k) = \{v_1 = 4, \ldots, v_n = 1\}$

- $dom(var(k))$ est l'espace des valeurs possibles pour les variables de $var(k)$. Il correspond à la zone possible ZP présentée dans la figure 2.3. On a $valvar(k) \in dom(var(k))$.
- $k^{\varepsilon} = k^{\varepsilon}(var(k))$ est le sous-espace de $dom(var(k))$ tel que k est satisfait. Il correspond à la zone de comportement normal *ZN* dans l'espace des valeurs des variables. On a $k^{\varepsilon} \subseteq dom(var(k))$.

Alors, dans l'espace des valeurs des variables (figure 2.3), si k est une contrainte décrivant $mode(I_i)$, on peut écrire :

$$mode(I_i) \leftrightarrow \forall valvar(k), valvar(k) \in k^{\varepsilon} \tag{2.1.3}$$

Dans l'espace des variables (figure 2.3), l'intersection entre les zones *ZN* et ZF_i, et parfois entre les zones ZF_i, montre qu'un jeu de valeurs $valvar(k)$ peut être consistant avec plusieurs modes d'un même composant.

Item non-modélisé : Un Item I_i est dit non-modélisé si aucune contrainte n'est disponible pour la description comportementale de cet Item. Un item non-modélisé a par défaut deux modes {ok, cfm }. Le mode *ok* d'un Item non-modélisé ne s'associe à aucune contrainte.

Dans certains travaux (De Kleer et Williams [1987]; Struss et Dressler [1989]; De Kleer et Williams [1992]; Chittaro et Ranon [2004]), les modes de comportement des composants sont définis via des contraintes. Cependant, ces auteurs n'ont pas abordé la relation entre les modes via l'espace des variables. En analysant les zones dans l'espace des valeurs des variables nous avons montré qu'un tuple de valeurs peut être consistant avec la description de plusieurs modes d'un même item. De plus, il peut exister dans la zone possible la partie qui est hors de ZN et associée à aucun mode spécifique. Ceci est la zone des défauts complémentaires non-spécifiée qui correspondent au mode *cfm* . Quand le modèle est construit au fur et à mesure, l'ensemble des modes de comportement d'un composant peut être défini partiellement. Un mode *cfm* peut être ajouté automatiquement pour représenter la zone des défauts complémentaires non-spécifiés.

2.1.4 Modèle élémentaire

Les éléments de base qui sont ajoutés dans le processus de diagnostic sont les contraintes. Le mode de comportement d'un item I_i est décrit par une ou plusieurs contraintes. Dans l'analyse diagnostique à base de modèle de consistance, la conception des tests de détection s'appuie sur les contraintes tandis que la formulation des résultats de test (test faut/test vrai) s'appuie sur des modes de comportement. Pour faire le lien entre ces deux aspects, nous avons introduit la notion de *modèle élémentaire*. Chaque modèle élémentaire est associé à une et une seule contrainte de comportement. Le modèle élémentaire est défini par

$$\langle mode(I_i), k, [K']) \rangle \tag{2.1.4}$$

avec

- $mode(I_i)$ est un mode de comportement/fonctionnement d'un item I_i qui est décrit par une ou plusieurs contraintes.
- k est une contrainte de comportement décrivant $mode(I_i)$.
- K' est un ensemble de contraintes de validité pour k.

  Un modèle élémentaire correspond à un seul mode mais comme plusieurs modèles élémentaires peuvent se rapporter à un même mode, un mode peut être décrit par plusieurs contraintes de comportement. Ceci est exprimé par la proposition (2.1.3).

  Nous allons analyser en détail la notion de *modèle élémentaire* en prenant en compte les contraintes de validité. Nous pouvons distinguer deux types de contraintes par :
  - Les contraintes de comportement $k(obs, V) = 0$ qui sont satisfaites pour des observations *obs* avec $V = var(k)$.
  - Une contrainte de validité $k'(obs, V') \diamond 0$ qui est satisfaite pour des observations *obs* avec $V' = var(k')$. L'opérateur $\diamond$ représente les opérateurs de comparaison comme $\leq$, $\geq$, $=$, ...

  Dans (Touaf et Ploix [2004b]), les contraintes de comportement et de validité sont modélisées par le tableau 2.1.

$k(obs,\ V) = 0$	$\bigwedge_i k'_i(obs,\ V'_i) \diamond 0$	$mode(I)$
faux	faux	vrai
faux	vrai	faux
vrai	faux	vrai
vrai	vrai	vrai

TAB. 2.1 – Vérité des modes selon les contraintes

Nous pouvons formuler à partir du tableau 2.1 :

$$mode(I) \leftrightarrow \forall obs \in OBS, (k(obs,V) = 0) \vee \neg(\bigwedge_i k'_i(obs, V'_i) \diamond 0) \tag{2.1.5}$$

En réalité, il est trop difficile de tester tous les $obs \in OBS$, (2.1.5) devient pratiquement

$$mode(I) \rightarrow \exists obs \in Obs, (k(obs,V) = 0) \vee \neg(\bigwedge_i k'_i(obs, V'_i) \diamond 0) \tag{2.1.6}$$

avec $Obs \subset OBS$

L'expression (2.1.6) montre qu'il est difficile de déterminer le mode actuel d'un item I_i. En revanche, si toutes les contraintes de validité $k'_i \in K'$ sont satisfaites et s'il existe une contrainte $k_i \in K$ qui n'est pas satisfaite par les observations, le mode correspondant peut être exclu. Il est exprimé par

$$(\exists obs \in Obs \setminus (k_i(obs, V_i) \neq 0) \wedge (\bigwedge_i k'(obs', V_i') \diamond 0)) \rightarrow \neg mode(I_i) \quad (2.1.7)$$

Il nous faut remarquer que 2.1.7 n'est pas un test car il contient des variables V_i et V_i' dont les valeurs sont inconnues. Les contraintes doivent être combinées entre elles pour éliminer les variables qui sont par nature inconnues.

Connexion entre les modèles élémentaires

Dans le problème de diagnostic avec interactions durant la phase de modélisation, le modèle du système est construit au fur et à mesure par l'opérateur. Les nouveaux modèles élémentaires ajoutés et les modèles existants sont connectés entre eux via des variables. Les variables peuvent être définies par des domaines de valeurs différents. Pour faire le lien entre les modèles élémentaires par des variables, on introduit la notion de connexion entre deux variables.

Supposons que I_i soit un item et que k_i soit une contrainte décrivant I_i. k_i fait intervenir des variables $var(k_i)$. Pour tout $v_i \in var(k_i)$, $dom(v_i)$ est l'ensemble de valeurs de v_i. Une connexion partielle est définie par :

Définition 2.1.5. *(Connexion partielle) Soient deux contraintes k_1 et k_2 avec $v_1 = v_{\angle dom_1} \in var(k_1)$, $v_2 = v_{\angle dom_2} \in var(k_2)$, $pcon(v_1 \rightarrow v_2, f)$ est la connexion partielle associée à la grandeur physique v entre la variable v_1 et la variable v_2 par la fonction non-inversible f si v_1 et v_2 se rapportent à la même grandeur physique et si pour chaque valeur de v_1, il existe une valeur de v_2 telle que $f(v_1) = v_2$.*

Une connexion complète est définie par :

Définition 2.1.6. *(Connexion complète) Soient deux contraintes k_1 et k_2 avec $v_1 = v_{\angle dom_1} \in var(k_1)$, $v_2 = v_{\angle dom_2} \in var(k_2)$, $fcon(v_1 \leftrightarrow v_2, f)$ est la connexion complète associée à la grandeur physique v entre la variable v_1 et la variable v_2 par la fonction f si v_1 et v_2 se rapportent à la même grandeur physique et si pour chaque valeur de v_1, il existe une valeur de v_2 tel que $f(v_1) = v_2$ et $f^{-1}(v_2) = v_1$.*

Pour simplifier, la connexion complète $fcon(v_1 \leftrightarrow v_2, f)$ est réécrite $fcon(v_1 \leftrightarrow v_2)$ car f est identique dans toutes les connexions complètes.

Représentation graphique de la connexion entre les modèles élémentaires via des variables :

- le cercle représente une grandeur physique.
- la flèche unidirectionnelle associée à la connexion représente une connexion partielle entre deux variables réalisée via une fonction f (Figure 2.4).

– la flèche bidirectionnelle associée à la connexion représente une connexion complète entre deux variables réalisée via une fonction f (Figure 2.5).
– le lien représente le lien entre une variable et sa grandeur physique.

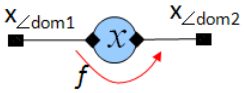


FIG. 2.4 – Connexion partielle

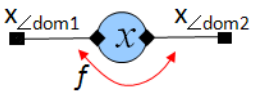


FIG. 2.5 – Connexion complète

Traitement des connexions entre les modèles élémentaires

Une fonction de connexion est considérée comme une contrainte représentant la relation entre deux variables d'une même grandeur physique. Pourtant, elle n'est associée à aucun mode de comportement. Une fonction de connexion est traitée comme une contrainte de comportement dans certaines tâches, par exemple la tâche de génération des SSTs (Dustegor *et al.* [2004]; Ploix *et al.* [2008]) ou la propagation des valeurs au sein des contraintes (annexe C),ect. Dans ces tâches, la notion de *variable déductible / non-déductible* (Ploix *et al.* [2008]) apparaissant dans une contrainte de comportement peut être considérée. Selon la définition 2.1.5, parmi deux variables d'une connexion partielle f, l'une est déductible et l'autre est non-déductible en considérant f.

Dans cette partie, nous avons introduit la notion de modèle élémentaire pour faire le lien entre la génération des tests de détection qui s'appuie sur les contraintes de comportement et la formulation des résultats de test qui s'appuie sur les modes. Un modèle élémentaire est considéré aussi comme un élément de base dans la construction du modèle d'abstraction d'un système parce que les niveaux d'abstraction entre des contraires doivent être liés avec les modes décrits par ces contraintes. Pour ajouter au fur et à mesure des modèles élémentaires dans la construction du système, les connexions doivent être définies. Nous avons distingué deux types de connexions : partielle et complète parce que les connexions peuvent être réalisées via des variables différents décrites sur différents domaines d'une même grandeur physique.

2.1.5 Formulation des symptômes

Un test de détection, aussi nommé *Relation Redondante Analytique (RRA)* dans la communauté FDI, est une relation mathématique entre des observations qui ne contient plus aucune variable. Un test consiste à tester les éléments impliqués dans un sous-système. Une fois que

les tests de détection sont donnés, la détection des défauts consiste à vérifier à chaque moment si ces RRAs sont consistants (les observations sont conformes à RRA) ou pas. Si un test est inconsistant (les observations sont non-conformes à RRA), un défaut est détecté, il s'agit un conflit entre les observations et le modèle de référence. Un test T_i peut être consistant (les observations sont conformes au modèle de référence) ou inconsistant (les observations sont non-conformes au modèle de référence). Les symboles $\perp(\dots),\top(\dots)$, qui indiquent respectivement la consistance ou l'inconsistance d'une contrainte dans (Apt [2003]), sont utilisés ici pour décrire les symptômes correspondant à un test. Notons un symptôme par :

- $\perp(T_i)$ si T_i est consistant (les observations sont conformes au modèle de référence implicite à T_i).
- $\top(T_i)$ si T_i est inconsistant (les observations sont non-conformes au modèle de référence implicite à T_i).

Dans la communauté DX, les tests consistants ne sont pas exploités dans le raisonnement diagnostique parce que le comportement dans le mode *ok* s'apparente très souvent à des modes de défauts qui sont partiellement non-bornés : la zone *ZN* intersecte généralement de nombreuses zones ZF_i. Ainsi, si un test T_i est consistant, il est impossible de conclure que tous les items impliqués dans le SST correspondant sont en mode de fonctionnement normal. Sinon, un test T_i est dit *complètement testé* (fully checked test). Nous allons détailler maintenant la formulation logique des symptômes en introduisant la notion d'*explication d'un test*.

Définition 2.1.7. *(Explication d'un test inconsistant) Explication E_i d'un symptôme $\top(T_i)$ est une clause (une disjonction logique de littéraux) dont chaque élément est un mode tel que pour toutes les observations si $\top(T_i)$ est vérifié alors E_i est vérifié. Il est écrit par $\forall obs/\top(T_i) \rightarrow E_i$, avec $E_i = Expl(\top(T_i))$.*

Exemple 3. Supposons que T_i soit un test de détection qui corresponde à un *SST*. T_i permet de tester les modes de fonctionnement normaux de l'ensemble $\{I_1, I_2, \dots, I_n\}$ impliqué. Supposons que T_i soit inconsistant, l'explication de $\top(T_i)$ est donnée par $\neg ok(I_1) \vee \neg ok(I_2) \vee \dots \vee \neg ok(I_n)$. On peut écrire

$$\forall obs/\top(T_i) \rightarrow \neg ok(I_1) \vee \neg ok(I_2) \vee \dots \vee \neg ok(I_n) \tag{2.1.8}$$

avec

$$E_i = Expl(\top(T_i)) = \neg ok(I_1) \vee \neg ok(I_2) \vee \dots \vee \neg ok(I_n) \tag{2.1.9}$$

Cet exemple peut être étendu à des modes de défaut spécifique. Par exemple :

$$\forall obs/\top(T_i) \rightarrow \neg mode(I_1) \vee \neg mode(I_2) \vee \dots \vee \neg mode(I_n) \tag{2.1.10}$$

Pour les tests consistants, nous allons distinguer les deux types : *test consistant normal* (appelé simplement *test consistant*) et *test consistant complètement testé*. Dans l'analyse diagnostique, nous ne nous intéressons qu'aux explications des tests qui sont complètement testés.

Un *test consistant normal* ne sera pas utilisé dans le raisonnement diagnostique mais sa signature sera exploitée pour classer les diagnostics. Ceci peut être consulté dans l'annexe A.

Définition 2.1.8. *(Explication d'un test consistant complètement testé) L'explication E_i d'un test consistant complètement testé $\perp(T_i)$ est un monôme (une conjonction logique de littéraux) dont chaque élément est un mode tel que pour toutes les observations si $\perp(T_i)$ est vérifié alors E_i est vérifié. Il est exprimé par $\forall obs/\perp(T_i) \rightarrow E_i$, notons $E_i = Expl(\perp(T_i))$.*

Exemple 4. Revenons à l'exemple (3). Supposons que T_i soit un test consistant et que T_i soit complètement testé. L'explication de $\perp(T_i)$ est donné par $Expl(\perp(T_i)) = ok(I_1) \wedge ok(I_2) \wedge \cdots \wedge ok(I_n)$. On peut écrire

$$\forall obs/\perp(T_i) \rightarrow ok(I_1) \wedge ok(I_2) \wedge \cdots \wedge ok(I_n). \tag{2.1.11}$$

avec

$$E_i = Expl(\perp(T_i)) = ok(I_1) \wedge ok(I_2) \wedge \cdots \wedge ok(I_n) \tag{2.1.12}$$

Dans cette partie, nous avons rappelé les éléments de formulation pour la phase de détection d'un problème de diagnostic : relation redondante analytique (RRA) ou test de détection, symptôme (un test consistant ou inconsistant) et l'explication d'un symptôme. Pour exploiter au mieux les résultats de tests, nous avons introduit la notion de *test consistant complètement testé*. En fait, l'hypothèse d'exonération venant de l'approche FDI (si un test est consistant alors il n'y a aucun défaut dans le sous-système testé, les composants impliqués dans ce sous-système sont exonérés) n'est pas toujours convenable dans l'approche bridge FDI et DX. En revanche, si tous les tests consistants sont ignorés dans l'analyse diagnostique, ce fait peut conduire à des diagnostics physiquement impossibles. Ce problème est abordé dans (Struss et Dressler [1989]; Desindes [2006]). Le fait d'un test consistant est complètement testé ou non est déterminé par l'expert à l'aide de sa connaissance implicite sur le système.

2.1.6 Conclusion

Dans cette partie, nous avons rappelé et souligné les éléments principaux d'un problème de diagnostic à base de modèle de consistance. Certaines notions ont été développées en visant à l'objectif de la formulation d'un modèle d'abstraction dans la suite.

2.2 Formulation de l'abstraction

Dans cette partie, nous souhaitons reformuler les modèles d'abstraction qui existent dans la littérature (abstraction structurelle, comportementale, etc.). Cette reformulation s'appuie sur une vision orientée objet afin de pouvoir concevoir un langage qui permette à l'expert

de décrire le système au fur et à mesure à différents niveaux d'abstraction. En se basant sur les éléments de base présentés précédemment, nous introduisons ensuite les notions suivantes : abstraction sur les variables, abstraction sur les modèles élémentaires, abstraction sur les items.

2.2.1 Abstraction sur les variables

La description au niveau comportemental se rapporte à des variables impliquées dans des contraintes de comportement. En effet, la précision de la description d'un phénomène physique, qui est caractérisé par la variable v, dépend du domaine de valeur $dom(v)$. Par exemple on note $v_{\angle dom_1}$ la variable v décrite dans le domaine de valeurs dom_1. De la même manière, on peut trouver une autre variable $v_{\angle dom2}$ telle que $v_{\angle dom2}$ est plus abstrait ou plus précise que $v_{\angle dom1}$. Cela correspond à la notion d'abstraction sur les variables.

Définition 2.2.1. *($v_{\angle dom_2}$ abstraction de $v_{\angle dom_1}$). Soient $v_{\angle dom_1}$ et $v_{\angle dom_2}$ deux variables associées à une même grandeur physique v, $v_{\angle dom_2}$ est l'abstraction de $v_{\angle dom_1}$ s'il existe une fonction non-inversible f telle que $\forall \vartheta_1 \in dom_1$, $\exists \vartheta_2 \in dom_2$ tel que $\vartheta_2 = f(\vartheta_1)$.*

Si f est une fonction inversible alors $dom_1 \equiv dom_2$. Dans ce cas, il n'y a pas d'abstraction sur les variables : les variables sont identiques.

Exemple 5. On prend l'exemple d'un circuit électrique contenant une lampe. Le comportement de la lampe peut s'expliquer par les deux grandeurs physiques $\mathscr{I}$, $\mathscr{L}$. $\mathscr{I}$ représente l'intensité circulant dans l'ampoule, $\mathscr{L}$ représente la quantité de la lumière en fonction de $\mathscr{I}$. On suppose que la variable $I_1 = I_{\angle dom_1}$ avec $dom_1 = \{[-30mA, -1mA] \cup [+1mA, +30mA], [-1mA, +1mA]\}$ est l'intensité nécessaire pour alimenter l'ampoule. Ce domaine de valeurs définit respectivement les intervalles d'intensité suffisante et insuffisante pour alimenter la lampe. Considérons une variable I_2 avec $dom(I_2) = \{0, 1\}$. Deux valeurs $\{0, 1\}$ correspondant respectivement à l'absence ou insuffisance de courant et à la présence du courant électrique dans le circuit. Il existe une relation non-inversible f qui associe à chaque valeur de I_1 une valeur de I_2.

$$I_2 = f(I_1) = \left\{ \begin{array}{c} 1 \; si \; 1mA \leq |I_1| \leq 30mA \\ 0 \; si \; |I_1| < 1mA \end{array} \right\}$$

Dans cet exemple, la grandeur physique $\mathscr{I}$ est caractérisée par deux partitions sous forme d'intervalles. I_2 est une abstraction de I_1.

L'abstraction sur les variables se rapporte aux définitions de *connexion partielle* et de *connexion complète* (définitions 2.4 et 2.5, page 33)

2.2.2 Abstraction sur les items

Les éléments principaux impliqués dans le modèle d'abstraction structurel présenté dans (Chittaro et Ranon [2004]) sont les *composants*. Pour la modélisation structuro-fonctionnelle FIS (présentée dans la section 1.3.2), le phénomène d'abstraction est étendu pour les fonctions et les ressources. Dans FIS, les ressources sont appelées de façon générique *item*. Dans (Chittaro et Ranon [2004]), le modèle d'abstraction est construit de bas en haut. Pour modéliser le système, Chittaro commence par le niveau détaillé des composants et construit le modèle d'abstraction niveau par niveau. Chaque niveau correspond à un nouveau problème de diagnostic plus ou moins abstrait. Dans notre problème, nous construisons le modèle au fur et à mesure de haut en bas. La décomposition peut être partielle. Cela veut dire que seuls certains items peuvent être décomposés à un moment donné. Le modèle d'un problème de diagnostic est un ensemble d'items décrits à différents niveaux d'abstraction. Concernant la décomposition partielle ou complète de l'expert, nous introduisons ici les notions d'*abstraction partielle* et d'*abstraction complète* des items. Ces définitions sont données en s'appuyant sur les modes des items. Pour cet objectif, nous introduisons tout d'abord la notion de *m-proposion* (proposition de mode).

Définition 2.2.2. *(m-proposition) Une proposition logique où les symboles sont des modes des* items*, qui peut être exprimée par une forme normale conjonctive ou par une forme normale disjonctive, est appelée une m-proposition. Si* $\mathscr{P}(mode_1, \ldots, mode_n)$ *est une m-proposition, le support de* $\mathscr{P}$ *est défini par* $Modes(\mathscr{P}) = \{mode_1, \ldots, mode_n\}$.

Exemple 6. $(mode_1 \rightarrow mode_2) \wedge mode_3$ avec $\neg mode_1 = mode_4 \vee mode_5$ est une m-proposition, parce qu'elle peut être réécrite : $(mode_4 \vee mode_5 \vee mode_2) \wedge mode_3$.

Le concept d'abstraction partielle peut être introduit.

Définition 2.2.3. *(abstraction partielle) Soient I un* item *et* $\mathbb{I} = \{I_1, \ldots, I_n\}$ *un ensemble d'items. I est une abstraction partielle de* $\mathbb{I}$ *si* $\forall m_i \in Modes(I)$ *et il existe une m-proposition* $\mathscr{P}_i$ *vrai satisfaisant* $m_i \rightarrow \mathscr{P}_i$ *avec* $Modes(\mathscr{P}_i) = \{mode(I_1), \ldots, mode(I_n)\}$.

Si I est une abstraction partielle de $\mathbb{I} = \{I_1, \ldots, I_n\}$, I est appelé *item-parent* de chaque *item* I_i. A l'inverse, chaque *item* I_i est appelé *item-enfant* de I. Normalement, si un *item-parent* se comporte normalement (mode *ok*), on en déduit que ses *items-enfants* sont aussi en mode normal. Cette propriété est représentée par une implication logique $ok(I) \rightarrow ok(I_1) \wedge ok(I_2) \wedge \cdots \wedge ok(I_n)$. Dans un contexte de coopération homme-automate, l'abstraction partielle représente la connaissance partielle de l'expert lors d'une décomposition du système. Si la décomposition est complète (tous les sous-items sont connus), nous avons une abstraction complète :

Définition 2.2.4. *(abstraction complète) Soient I un* item *et $\mathbb{I} = \{I_1, \ldots, I_n\}$ un ensemble d'items. I est une abstraction complète de $\mathbb{I}$ si $\forall m_i \in Modes(I)$ et il existe une m-proposition $\mathscr{P}_i$ vrai satisfaisant $m_i \leftrightarrow \mathscr{P}_i$ avec $Modes(\mathscr{P}_i) = \{mode(I_1), \ldots, mode(I_n)\}$.*

Notion d'item virtuel

Une abstraction partielle $\mathbb{I} = \{I_1, \ldots, I_n\}$ de I peut toujours être transformée en une abstraction complète en introduisant un *item virtuel* qui représente la partie complémentaire de I qui n'est pas incluse dans $\mathbb{I}$, notée par VI pour *item virtuelle*, avec $VI = I \setminus \mathbb{I}$. VI est un *item non-explicité*. Bien qu'un tel item ne soit pas explicité, on peut concevoir, au minimum, deux modes *ok* et *cfm* . Notons que le mode *ok* d'un item non-explicité n'est associé à aucune contrainte.

Pour le modèle de fonctionnement normal où seulement deux modes (*ok* , *cfm*) sont disponibles pour chaque item, si un item virtuel VI est ajouté dans l'ensemble non-complet de sous-items $\{I_1, \ldots, I_n\}$ d'une super-item I, nous avons :

$$ok(I) \leftrightarrow ok(I_1) \wedge \cdots \wedge ok(I_n) \cdots \wedge ok(VI) \tag{2.2.1}$$

Exemple 7. Par exemple, *voiture* peut être décomposé partiellement en *Système d'Allumage (SA)* et en *Système de Démarrage (SD)* (abstraction partielle). Les modes *{ok,cfm}* sont ajoutés automatiquement pour les items : $Modes(SA) = \{ok, cfm\}$, $Modes(SD) = \{ok, cfm\}$. Un item virtuel $VI = voiture \setminus \{SA, SD\}$, qui représente la partie complémentaire de *SA* et *SD* dans *voiture*, peut être ajouté avec $Modes(VI) = \{ok, cfm\}$. Une abstraction complète est obtenue. Elle est décrite par la proposition 2.2.2.

$$\begin{aligned} ok(SA) \wedge ok(SD) \wedge ok(VI) &\rightarrow ok(voiture) \\ cfm(SA) \vee cfm(SD) \vee cfm(VI) &\rightarrow cfm(voiture) \end{aligned} \tag{2.2.2}$$

Des modes de défaut spécifiques peuvent aussi être introduits par l'expert. Par exemple les modes créés peuvent être $Modes(voiture) = \{ok, no-start, cfm\}$, $Modes(SA) = \{ok, no-spark, cfm\}$, $Modes(SD) = \{ok, no-crankcfm\}$. Les relations causales sont définies par l'expert : $no-spark(SA) \rightarrow no-start(voiture)$, $no-crank(SD) \rightarrow no-start(voiture)$. La description partielle du système est maintenant :

$$ok(voiture) \rightarrow ok(SA) \wedge ok(SD) \tag{2.2.3}$$

$$no-spark(SA) \rightarrow no-start(voiture) \tag{2.2.4}$$

$$no-crank(SD) \rightarrow no-start(voiture) \tag{2.2.5}$$

Notons que l'abstraction de $\{SA, SD\}$ de *voiture* est une abstraction partielle. La proposition 2.2.3 est donc déduite.

Les propositions logiques sont complétées par les modes :

$$ok(SA) \wedge ok(SD) \wedge ok(VI) \rightarrow ok(voiture) \tag{2.2.6}$$
$$no-spark(SA) \wedge ok(SD) \wedge ok(VI) \rightarrow no-start(voiture) \tag{2.2.7}$$
$$ok(SA) \wedge no-crank(SD) \wedge ok(VI) \rightarrow no-start(voiture) \tag{2.2.8}$$
$$cfm(SA) \vee cfm(SD) \vee cfm(VI) \rightarrow cfm(voiture) \tag{2.2.9}$$

Les implications logiques sont alors rassemblées pour obtenir des équivalences logiques. Par exemple, à partir de 2.2.3 et 2.2.5, une équivalence logique est obtenue :

$$ok(voiture) \leftrightarrow ok(SA) \wedge ok(SD) \wedge ok(VI) \tag{2.2.10}$$

Supposons que l'information donnée par l'expert sur le mode $no-start(voiture)$ soit complète et qu'il n'y ait pas d'autres cas qui conduisent au mode $no-start(voiture)$. Comme il n'y a que deux propositions (2.2.7) et (2.2.8) qui représentent l'implication d'une conjonction des modes enfant à un mode parent mode *no-start(voiture)*, une équivalence logique est obtenue :

$$\begin{aligned} no-start(voiture) \leftrightarrow ((no-spark(SA) \wedge ok(SD)) \vee \\ (no-crank(SD) \wedge ok(SA))) \wedge ok(VI) \end{aligned} \tag{2.2.11}$$

En ignorant les modes *ok* dans la proposition 2.2.11, nous avons :

$$no-start(voiture) \leftrightarrow (no-spark(SA) \vee no-crank(SD)) \tag{2.2.12}$$

De 2.2.9, nous déduisons :

$$cfm(voiture) \leftrightarrow cfm(SA) \vee cfm(SD) \vee cfm(VI) \tag{2.2.13}$$

Représentation graphique de l'abstraction sur les items

Graphiquement, l'abstraction entre les items est représentée par un graphe, dont

- chaque noeud est un item.
- chaque arc représente un lien entre un item enfant et un item parent. Pour distinguer ce type de lien d'autres, l'extrémité côté item parent est un rond. Pour distinguer, le lien parent-enfant dans une abstraction partielle, on utilise un arc en pointillé (figure 2.6), alors que dans une abstraction complète, on utilise un arc normal (figure 2.7).

Exemple 8. Prenons l'exemple de *voiture* de l'exemple 7. *la voiture* est décomposée partiellement en sous-items : *système de démarrage (SD)* et *système d'allumage (SA)*. L'abstraction partielle et l'abstraction complète sont représentées respectivement sur les figures (2.8) et (2.9).

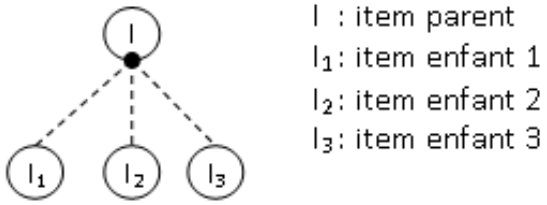


FIG. 2.6 – Abstraction partielle sur les items

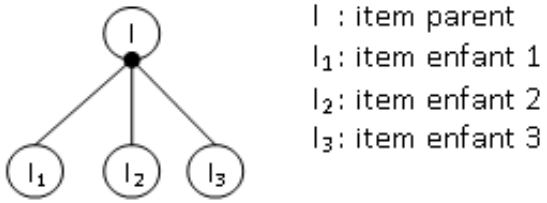


FIG. 2.7 – Abstraction complète sur les items

2.2.3 Abstraction sur les modèles élémentaires

Un problème qui peut apparaître quand les modèles élémentaires sont ajoutés au fur et à mesure est l'existence de relations d'abstraction entre les modèles élémentaires. Comme les modèles élémentaires décrivent le modèle comportemental des items, l'abstraction entre les modèles élémentaires se rapporte à la notion d'abstraction sur les items. Dans cette partie, nous donnons une formalisation de l'abstraction sur les modèles élémentaires afin de l'exploiter plus tard dans l'analyse diagnostique.

Définition 2.2.5. *(Abstraction sur les modèles élémentaires).*

Soient $\mathscr{K}$ un ensemble de modèles élémentaires qui comporte des contraintes (contraintes de comportement et de validité) $\mathbb{K}$ décrivant le mode $mode(I_i)$ et $\mathscr{K}'$ un ensemble de modèles élémentaires qui comporte des contraintes (contraintes de comportement et de validité) $\mathbb{K}'$ décrivant les modes $\{mode_1(I'_1),\ldots,mode_n(I'_n)\}$. $\mathbb{K}^{\varepsilon}$ et $\mathbb{K}'^{\varepsilon}$ sont respectivement des zones dans les espaces des valeurs des variables où $\mathbb{K}$ et $\mathbb{K}'$ sont satisfaits. $\mathscr{K}$ est une abstraction de $\mathscr{K}'$ si et seulement si :

$$mode_1(I'_1) \wedge \cdots \wedge mode_n(I'_n) \rightarrow mode(I_i) \tag{2.2.14}$$

$$\exists \textit{ une projection } \Pi/\Pi\left(\mathbb{K}'^{\varepsilon}\right) = \mathbb{K}^{\varepsilon} \tag{2.2.15}$$

La proposition 2.2.14 montre que l'abstraction sur les modèles élémentaires se rapporte à l'abstraction sur les items donnée par la définition 2.2.4, page 38. L'abstraction sur les modèles élémentaires existe implicitement dans l'abstraction entre les items.

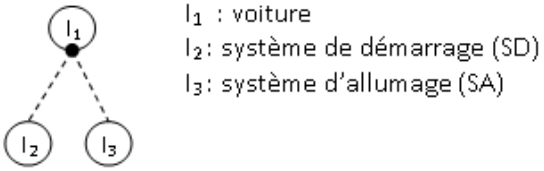


FIG. 2.8 – Abstraction partielle

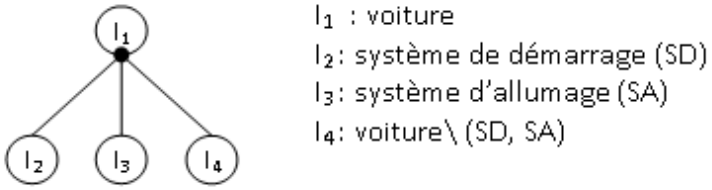


FIG. 2.9 – Abstraction complète

La proposition 2.2.15 peut être exprimée dans l'espace des variables (par exemple l'espace des variables présentée sur la figure 2.10). Nous voyons que dans l'abstraction sur les modèles élémentaires, les cas suivants peuvent apparaître :

- Au niveau le plus détaillé, de nouvelles grandeurs physiques peuvent apparaître. Elles permettent de décrire plus en détail le comportement des items. Ceci est exprimé par $VAR(\mathbb{K}) \subseteq VAR(\mathbb{K}')$ avec $VAR(\mathbb{K})$ et $VAR(\mathbb{K}')$ respectivement l'ensemble des grandeurs physiques impliquées dans $\mathbb{K}$ et $\mathbb{K}'$.
- L'abstraction entre les variables impliquées dans $var(\mathbb{K})$ et $var(\mathbb{K}')$.
  $valvar(\mathbb{K},\mathbb{K}')$ est un jeu de valeurs des variables impliquées dans $var(\mathbb{K},\mathbb{K}')$, chaque jeu de valeurs $valvar(\mathbb{K},\mathbb{K}')$ correspond à une observation *obs* sur l'ensemble des grandeurs physiques $VAR(\mathbb{K},\mathbb{K}')$ (voir page 25).

En faisant le lien entre la définition 2.2.5 et les définitions 2.2.3, 2.2.4 sur l'abstraction des items, nous voyons que ces notions peuvent être expliquées via des modèles élémentaires (propositions 2.2.14, 2.2.15) une fois que les modèles comportementaux sont ajoutés pour les items. Autrement dit, le phénomène d'abstraction sur les modèles élémentaires existe implicitement dans le phénomène d'abstraction sur les items. Ce fait est exprimé par la proposition 2.2.14.

Propriété 1. Soient $\mathscr{K} = \bigcap_{i=1\ldots n}(mode(I),k_i,K'_i)$ et $\mathscr{K}' = \bigcap_{j=1\ldots n'}(mode(I'),k_j,K'_j)$, deux ensembles de modèles élémentaires décrivant respectivement le comportement de I et de I'. Si I est un sous-item de I' dans une relation d'abstraction sur les items, $\mathscr{K}$ appartient nécessairement à l'ensemble des sous-modèles élémentaires de $\mathscr{K}'$.

Dans certains cas particuliers, nous pouvons avoir une abstraction entre les modèles élémentaires associés à un même item. Au cours des interactions du processus de diagnostic,

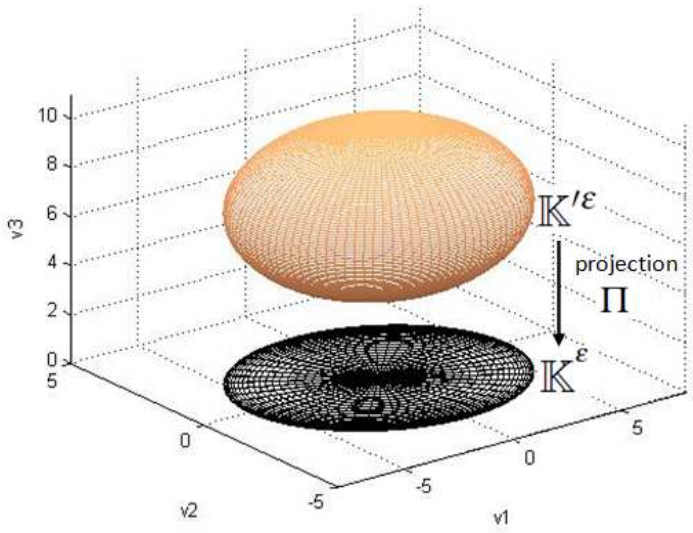


FIG. 2.10 – Espace des variables

l'opérateur peut affiner un modèle existant d'un item par un autre modèle plus détaillé. De nouvelles variables peuvent apparaître à ce niveau.

Exemple 9. Revenons à l'exemple du bac (exemple 2). Supposons qu'au début, le modèle qualitatif du système soit donné par :

$$(ok(bac),\ k_1) \tag{2.2.16}$$

avec k_1 donné sous forme d'une table :

Modes	q_{i_1}	q_{o_1}
ok	présent	présent
ok	absent	absent

TAB. 2.2 – Table de comportement de k_1

avec q_{i_1} et q_{o_1} sont respectivement les débits à l'entrée et à la sortie du bac, $dom(q_{i_1}) = dom(q_{o_1}) = \{présent,\ absent\}$.

Supposons qu'à un moment donné du processus de diagnostic, le modèle comportemental du bac soit enrichi par les modèles continus suivant :

$$\begin{aligned} q_{i_1\angle\{présent,absent\}} &\longrightarrow q_{i_1\angle R} \\ q_{o_1\angle\{présent,absent\}} &\longrightarrow q_{o_1\angle R} \\ (k_2 : S\frac{dh_1}{dt} &= q_{i_1} - q_{o_1},\ ok(bac)) \\ (k_3 : q_{o_1} &= kh_1,\ ok(bac)) \end{aligned} \tag{2.2.17}$$

Nous voyons que, selon la définition 2.2.5, $(ok(bac),\ k_1)$ est l'abstraction sur les modèles élémentaires $(ok(bac),\ k_2)$, $(ok(bac),\ k_3)$. Au niveau du sous-modèle, une nouvelle variable h_1 apparaît qui permet de tester plus en détail le système. Cet exemple montre le cas où l'abstraction sur les modèles élémentaires se produit au niveau d'un même item.

Représentation graphique de l'abstraction sur les modèles élémentaires

Graphiquement, l'abstraction entre les modèles élémentaires est représentée par un graphe (figure 2.11), dont

- chaque noeud est un modèle élémentaire.
- chaque arc représente un lien entre un sous-modèle élémentaire et un super-modèle élémentaire. Comme l'abstraction sur les items, l'extrémité du côté du super-modèle élémentaire est un rond.
  En comparaison avec la notion d'abstraction sur les items, nous ne distinguons pas l'abstraction partielle et complète au niveau des modèles élémentaires.

La figure 2.11 illustre d'un graphe représentant l'abstraction sur les modèles élémentaires. $\{(ok(I_1),k_1)\}$ est le modèle abstrait de $\{(ok(I_2),k_2),\ (ok(I_3),k_3),\ (ok(I_4),k_4)\}$.

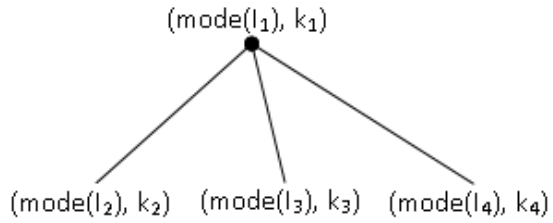


FIG. 2.11 – Abstraction partielle sur les modèles élémentaires

La compensation des défauts au niveau des modèles élémentaires

Supposons que $(mode(I_i),\ k_i)$ est l'abstraction sur les modèles élémentaires de $\mathbb{K} = \{(mode(I_1),\ k_1),...,(mode(I_n),\ k_n)\}$. Au niveau des contraintes, la compensation des défauts est représentée par le fait qu'il existe une observation telle que deux ou plus de deux contraintes impliquées dans $\mathbb{K}$ sont insatisfaites tandis que k_i est satisfait.

Exemple 10. On prend l'exemple de deux inverseurs C_1 et C_2 connectés en série :

Supposons que $(ok(C_1),k_1)$ et $(ok(C_2),k_2)$ soient respectivement deux modèles élémentaires qui décrivent les items C_1 et C_2 avec $k_1 = \{x_2 = \neg x_1\}$, $k_2 = \{x_3 = \neg x_2\}$, $dom(x_1) = \{0,1\}$, $dom(x_2) = \{0,1\}$, $dom(x_3) = \{0,1\}$. Supposons que $(ok(C),k)$ est l'abstraction de $\{(ok(C_1),k_1),(ok(C_2),k_2)\}$. On constate que si les deux inverseurs C_1 et C_2 sont en défaut en même temps tel que la valeur de sortie est égale à la valeur d'entrée, k_1 et k_2 ne sont pas satisfaits alors que k l'est. Les défauts de C_1 et C_2 sont compensés en terme de contraintes.

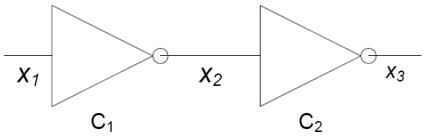


FIG. 2.12 – Deux inverseurs connectés en séries

2.2.4 Conclusion

Dans cette partie, nous avons reformulé le modèle d'abstraction d'un système. En comparaison avec les travaux existant dans la littérature, cette formulation s'appuie sur un point de vue orienté-objet. Le modèle hiérarchique est défini via les éléments de base du système : variable, modèle élémentaire et item (fonction ou ressource). Comme le modèle du système est construit au fur et à mesure à différents niveaux de détail, cette formulation vise à la conception d'un outil d'aide au diagnostic qui permette à l'expert de décrire le système au fur et à mesure et d'expliciter ses connaissances sur le système.

2.3 Modélisation structuro-fonctionnelle

Pour décrire un système à différents niveaux de détails, la modélisation fonctionnelle paraît être un outil intéressant. Le fonctionnement d'un système s'exprime parfois plus facilement par des fonctions que par des ressources. Mais le niveau fonctionnel doit être relié au niveau structuro-comportemental. Des ressources peuvent être associées à chaque fonction au moyen de différentes relations. Une modélisation structuro-fonctionnelle est rappelée dans ce chapitre.

2.3.1 Méthode FIS

La méthode proposée est le fruit d'une démarche de formalisation, appelée FIS, qui a commencé il y a plusieurs années (Flaus et Grandamas [2002]; Flaus [2003]; Froquet et Flaus [2003]; Flaus [2008]) et qui vise à être la plus complète possible tout en évitant l'écueil d'une trop grande lourdeur la rendant inapplicable. Les applications successives à des cas réels, techniques et organisationnels (Flaus [2010]), et l'intégration de cette méthode de modélisation à une démarche plus large incluant le diagnostic (Desinde *et al.* [2006]) et même la simulation (Thevenon et Flaus [2000]) a permis d'aboutir à une méthodologie pertinente. Elle a donné lieu au développement d'un outil logiciel (XRisk).

Nous résumons ici le principe de la méthode FIS, présenté en détail dans (Flaus [2008]). L'élément de base de la représentation est le *processus* ou *système*, composé de *fonctions*, de

ressources physiques, et de *ports d'entrées/sorties* et de *supports* permettant des *interactions* entre les systèmes (figure 2.13). Définissons ces différents termes.

Définition 2.3.1. *(norme ISO 9001) : Un processus ou système est un ensemble organisé d'activités qui utilise des ressources (personnel, équipement, matériels et machines, matière première et informations) pour transformer des éléments entrants en éléments de sortie dont le résultat final attendu est un produit.*

Définition 2.3.2. *(ressource)*

Les ressources sont constituées d'éléments matériels (machines et matières), humains et organisationnels permettant de mener les activités du processus. On notera que les méthodes (procédures, modes opératoires, programmes, etc.) font partie des ressources. Les ressources peuvent être caractérisées par un ensemble de variables, et leur comportement peut être décrit par un modèle de comportement, qui est utile pour le diagnostic.

Définition 2.3.3. *(fonction)*

Les fonctions du système sont définies comme le rôle d'un ensemble de ressources exprimé en terme de finalité. C'est une sorte de dématérialisation d'un ensemble d'entités indiquant un rôle, une action : ce qu'il peut faire. Elle représente une spécification du comportement du système. Une fonction peut être active ou non-active à un instant donné. Son comportement est caractérisé par un ensemble de variables, et éventuellement un modèle de comportement. Chaque fonction consomme ou utilise des ressources : ce sont ses ressources d'entrée (l'ensemble de celles-ci est noté $R_{in}(f)$). Inversement une fonction agit sur ou produit des ressources : ce sont ses ressources de sortie (ensemble noté $R_{out}(f)$).

Définition 2.3.4. *(entrant)*

Les entrants sont des éléments utilisés par le processus. Dans l'approche FIS, les entrants sont analysés en identifiant les ressources des autres systèmes requises par les fonctions du processus et qui sont fournies par les autres processus.

Définition 2.3.5. *(sortant)*

Les sortants sont les éléments générés par le processus. Ils se traduisent par des flux de ressources produites ou des actions sur des ressources présentes dans d'autres systèmes.

Définition 2.3.6. *(supports des ressources)*

Les supports des ressources sont une classe spécifique d'entrants requis pour faire en sorte que les ressources soient en état de bon fonctionnement. Les processus supports sont les processus générant comme sortants ces services supports.

Les ressources et fonctions, appelées de façon générique *items*, peuvent, à l'intérieur d'un même système être décomposées en sous items, et ainsi de suite. En ce qui concerne les fonctions, on peut définir des relations parent-enfant entre les fonctions du système, faisant émerger des fonctions nouvelles de haut niveau à partir d'un ensemble de fonctions élémentaires.

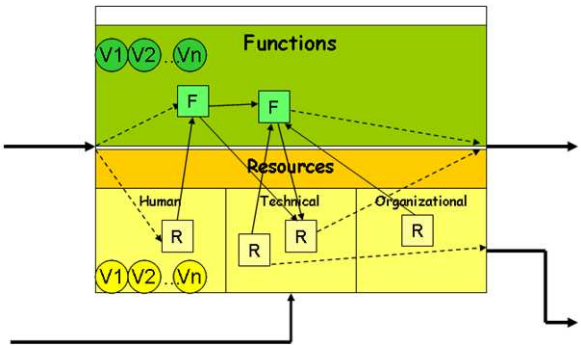


FIG. 2.13 – Modèle d'un système

En ce qui concerne les ressources, on peut aussi définir des relations parent-enfant permettant une abstraction matérielle du système (figure 2.14). Par exemple, les machines pourront être les enfants de la ressource atelier, les opérateurs enfants de la ressource personnel, etc. On fait apparaître ainsi les hiérarchies réelles du système en parallèle avec la décomposition fonctionnelle.

Un système peut alors être vu comme une "boîte", qui contient les ressources et les fonctions (2.13). Il ne possède pas d'attributs en propre et marque les limites des fonctions et ressources. Un système est décrit par l'ensemble des variables, des fonctions et des ressources. Ses modes de défaillance sont ceux des fonctions et des ressources. Il a pour but de définir un ensemble d'éléments qui interagissent et qui seront analysés ensemble. Par ailleurs, il joue un rôle vis-à-vis de la visibilité externe des éléments fonctionnels et matériels :

– dans les relations inter-systèmes, seuls les éléments connectés en interne aux *entrants/sortants* sont "vus" de l'extérieur.
– dans la décomposition hiérarchique *système/sous système*, seuls sont vus du parent les ressources de plus haut niveau et les fonctions définies comme visibles. Cela permet d'avoir une vue systémique du système

Pour les fonctions et les ressources, on définit une liste de modes de défaillance ou défauts, qui sont des étiquettes identifiant des modes de fonctionnement hors spécifications. L'ensemble de ces modes ajoutés au mode de bon fonctionnement forme l'ensemble des modes de fonctionnement de l'entité. Chacun de ces modes peut être actif ou non (état booléen).

Une ressource peut être partagée entre plusieurs systèmes, ce qui signifie qu'il doit exister une interaction physique entre ces systèmes de façon à permettre un transfert de ressource entre système (soit physiquement, soit virtuellement via une prise de contrôle). De la même façon, une fonction peut être partagée.

Les systèmes analysés pouvant devenir rapidement complexes, il est utile de découper le système analysé en sous-systèmes, qui seront éventuellement redécomposés. Les sous-fonctions et/ou sous-ressources peuvent être regroupées dans des sous-systèmes. Une double

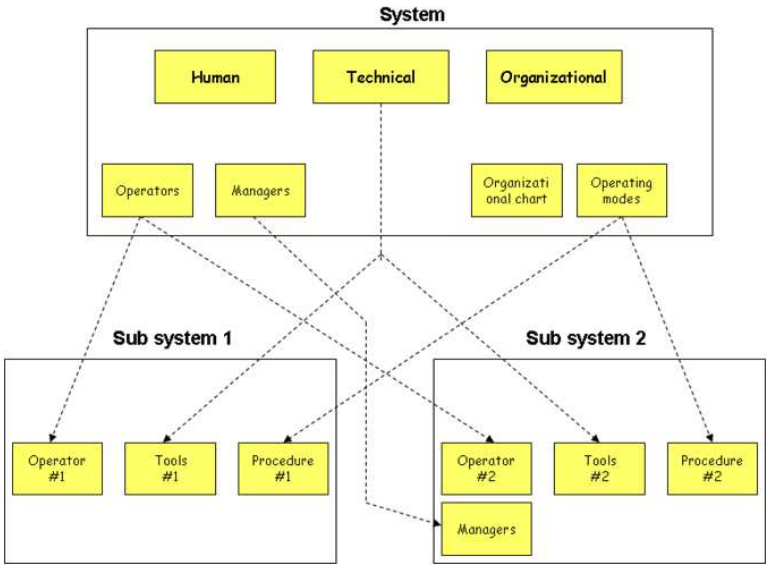


FIG. 2.14 – Modèle d'un système

hiérarchie fonctionnelle et matérielle apparaît.

Le modèle FIS permet aussi de définir les liens dysfonctionnels entre les modes de défaillances des items :

- Par défaut, si le parent est en bon fonctionnement (aucun mode de défaillance actif), cela impose qu'il en est de même des enfants et inversement. Il est possible de spécifier des relations plus complexes.
- Les modes de défaillances des fonctions peuvent être exprimés comme une relation logique entre des modes de défaillances des ressources d'entrée.
- Les modes de défaillances des ressources peuvent être exprimés comme une relation logique entre des modes de défaillances des fonctions pour lesquelles elles sont en sortie.

2.3.2 Conclusion

Nous avons rappelé de façon détaillée dans cette partie le modèle FIS original. Ceci donne un outil pour décrire itérativement le système dans le contexte du problème de diagnostic itératif. En plus des travaux présentés dans (Ploix et Chazot [2006]), le modèle FIS permet de représenter le modèle du système non seulement au niveau structurel mais aussi au niveau fonctionnel. Il permet donc d'expliciter mieux l'expertise de façon plus claire dans le processus de diagnostic. De plus, FIS n'exige pas des informations sur le modèle comportemental du système.

2.4 Modélisation interactive dans l'analyse diagnostique en utilisant la modélisation structuro-fonctionnelle FIS

Jusqu'ici, nous avons présenté les éléments de formulation d'un problème de diagnostic dans les sections 2.1 et 2.2, et le modèle structuro-fonctionnel FIS dans la section 2.3. Nous allons montrer dans cette partie comment rapprocher ces deux aspects dans le contexte d'un problème de diagnostic itératif.

L'idée principale est qu'à partir du modèle FIS, nous pouvons réaliser une analyse de risque puis une analyse diagnostique. Pour utiliser la modélisation FIS pour le diagnostic, il est nécessaire de définir les relations d'observations. Par ailleurs, le modèle dysfonctionnel doit être décrit. Une approche associant une variable à chaque mode est proposée dans (Flaus et Pham [2010]). Nous présentons ici une approche qui partitionne l'ensemble des modes en état possible recherché et symptôme. puis une proposition pour générer certains éléments du modèle dysfonctionnel pour le module d'analyse diagnostique.

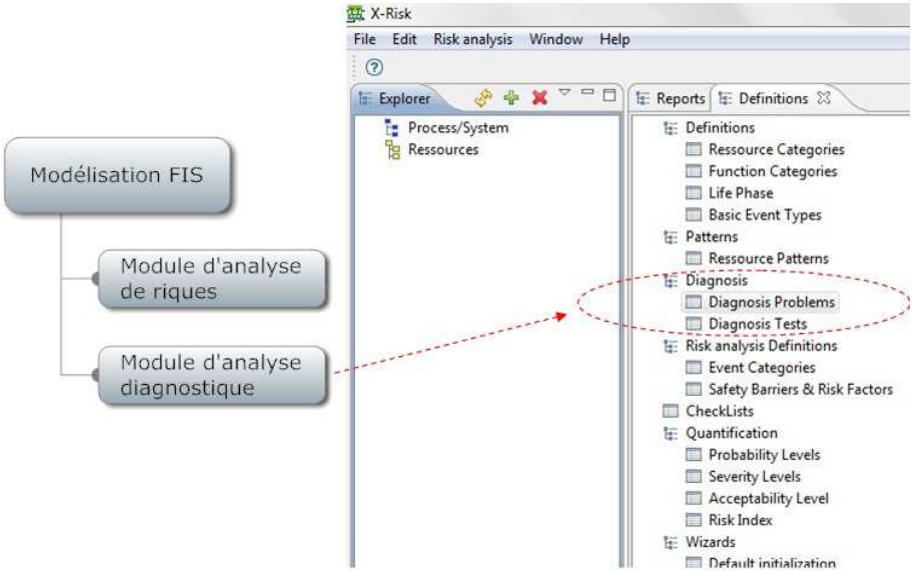


FIG. 2.15 – Module d'analyse diagnostique intégré dans XRisk

2.4.1 Distinction entre mode pour le diagnostic (d-mode) et symptôme

Dans la communauté de l'analyse diagnostique DX et FDI, une ressource est définie comme une entité pour laquelle on s'interroge sur son état de marche : c'est ce que l'on souhaite potentiellement réparer, corriger. Dans le modèle FIS, la notion de ressource est définie au sens plus large. Les ressources peuvent être de différentes natures : ressources matérielles, ressources fonctionnelles, ressources humaines, ressources informelles ou des ressources de type flux ou potentiel (chaleur, courant électrique, flux d'eau...). Pour cette raison, la notion

de mode est définie dans un sens plus large dans FIS qu'en diagnostic. Pour intégrer la notion de mode pour le diagnostic présenté dans les sections 2.2 et 2.1 dans le modèle FIS, nous proposons de distinguer les modes existants dans FIS en deux types :

- *d-mode (diagnostic mode)* : le mode d'une ressource dans FIS est défini comme *d-mode* dans le module de diagnostic itératif (intégré dans X-Risk) quand il correspond à un état possible recherché dans l'analyse diagnostique. Les d-modes correspondent à 2.1.2, page 29. Par exemple *ok(moteur)* est un *d-mode*.
- *symptôme* : le mode d'une ressource dans FIS peut aussi être défini comme un *symptôme* dans le module de diagnostic itératif quand il correspond à une appréciation de conformité de l'expert. Selon la section 2.1.5, un *symptôme* peut avoir deux états *{conforme, non-conforme}* (conforme ou non conforme à une référence attendue). Par exemple, $\perp_{ref}$*(lumière)* est un *symptôme* d'une ressource *lumière*. $\perp_{ref}$*(lumière)* indique que le *symptôme* est en état *conforme*. En d'autres termes, lorsqu'une fonction est réalisée, la lumière est conforme à la référence attendue. Le niveau de conformité peut être exprimé par le dégré de satisfaction grâce à la logique floue. Ceci est présenté en détail dans le chapitre 5.

Notons qu'un mode d'une fonction est toujours un *d-mode* parce qu'il n'est pas un phénomène physique observable : cela correspond à un élément possible d'un diagnostic.

Pour le module d'aide au diagnostic, nous allons associer chaque ressource à *une case à cocher*. Dans le modèle FIS, la case d'une ressource est cochée si ses modes correspondent à des symptômes. Si une ressource est cochée, deux symptômes *{conforme, non-conforme}* sont ajoutés automatiquement à cette ressource. Au contraire, si une ressource n'est pas cochée, deux d-modes *{ok, cfm}* sont ajoutés automatiquement pour la ressource.

La case à cocher est considérée comme l'information complémentaire que l'expert ajoute au modèle du système. Ceci est un type d'information qu'il n'est pas pertinent de formuler a priori car cela ajouterait une lourdeur dans la modélisation fonctionnelle qui n'existe pas aujourd'hui. Cette information n'est utile que lorsqu'il s'agit de calculer des diagnostics. Par sa connaissance implicite, l'expert peut distinguer en utilisant *une case à cocher*, au moment de faire appel au module de diagnostic.

2.4.2 Analyse des types de relation entre items et modes

Après avoir analysé en détail les modes des ressources dans le modèle FIS, nous allons analyser les relations logiques qui peuvent exister entre les modes des items dans FIS. L'idée est de montrer la possibilité de transformer le modèle FIS décrit par l'expert en un ensemble de propositions logiques exploitables dans l'analyse diagnostique. Cette étape de transformation est illustrée sur la figure 2.16.

Les ressources et fonctions peuvent être appelées de façon générique *items*. La procédure

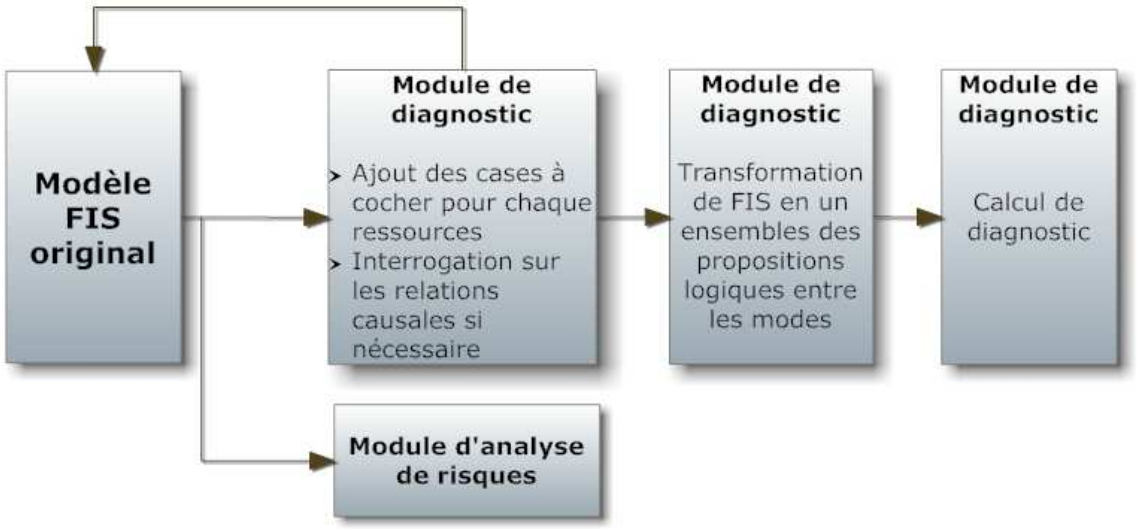


FIG. 2.16 – Module d'analyse diagnostique en s'appuyant sur FIS

de décomposition du système commence souvent par l'identification de l'item globale du système, puis on le décompose en *sous-items*. La décomposition est un processus itératif. En faisant le lien avec FIS (figure 2.17), nous voyons l'apparition des types de lien suivants :

- les relations d'abstraction entre les items : le lien numéro (1) sur la figure 2.17. Nous pouvons distinguer ici les deux types :
  - les relations de sous-fonctions/super-fonction
  - les relations de sous-ressources/super-ressource
- les ressources d'entrée d'une fonction (lien (2) sur la figure 2.17)
- les ressources de sortie d'une fonction (lien (3) sur la figure 2.17)

Nous allons détailler chaque type de relation.

1. Relations sous-items/item-parent (sous-ressources/ressource-parent, sous-fonctions/ fonction-parent) (liens (1) sur la figure 2.17)

   Une relation *sous-items/item-parent* est une abstraction sur les items que nous avons discutée dans la rubrique 2.2.2. Cela correspond à la relation hiérarchique présentée dans FIS. Structurellement, ceci est exprimé par le fait que la composition structurelle d'un ensemble de sous-items conduit à un item-parent. La composition peut être partielle ou complète. Une composition est partielle si seulement une partie des sous-items de l'item parent sont connus. Elle est complète si tous les sous-items de l'item-parent sont connus. Nous pouvons distinguer deux types de relations : sous-ressources/ressource-parent, sous-fonctions/fonction-parent.

   Prenons ici un exemple de relation *sous-ressources/ressource-parent* : *voiture* est une ressource matérielle, elle peut être décomposée en sous-ressources comme *roues*, *embrayage*, *démarreur électrique*, *moteur* . . .*voiture* est dite une ressource-parent dans cette

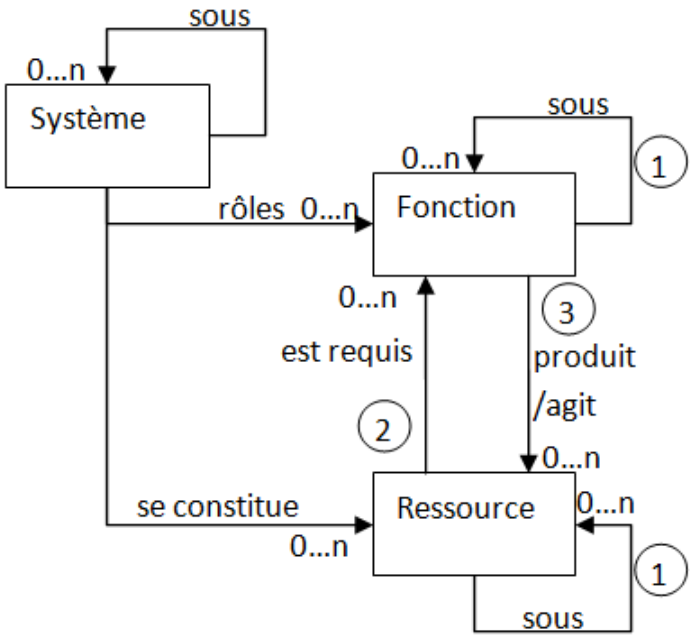


FIG. 2.17 – Présentation UML du modèle FIS

composition partielle.

En terme de mode de comportement, si un item-parent est en mode de fonctionnement normal (mode *ok*), il est évident que ses sous-items sont aussi en mode de fonctionnement normal. Supposons que I soit un item-parent et $\{I_1, \ldots, I_n\}$ soit l'ensemble de sous-items de I, nous avons deux cas :

– Si $\{I_1, \ldots, I_n\}$ est une décomposition partielle de $\mathbb{I}$, nous avons :

$$ok(I) \rightarrow ok\,(I_1) \wedge \cdots \wedge ok\,(I_n) \tag{2.4.1}$$

– Dans le cas où $\{I_1, \ldots, I_n\}$ est une décomposition complète de $\mathbb{I}$, nous avons :

$$ok(I) \leftrightarrow ok\,(I_1) \wedge \cdots \wedge ok\,(I_n) \tag{2.4.2}$$

Des relations plus complexes peuvent être décrites si besoin. Dans ce cas, les modes de défaut de I peuvent toujours être expliqués via des modes de sous-items. Ceci est détaillé dans la rubrique 2.2.2, page 37.

Remarque : Les relations *sous-ressources / ressouce-parent* ne sont définies que pour les ressources dans FIS dont les modes sont les d-modes dans le module d'analyse diagnostique. Nous ne nous intéressons pas à l'abstraction des ressources dans FIS dont les modes sont des symptômes dans le module d'analyse diagnostique.

2. Relations entre ressources d'entrée et fonctions (lien (2) sur la figure 2.17)

   Nous allons ici considérer deux cas : ressources d'entrée dans FIS dont les modes sont les *d-modes* ou les *symptômes* dans le module d'analyse diagnostique.

– Ressources d'entrée dans FIS dont les modes sont les *d-modes* dans le module d'analyse diagnostique.
L'ensemble des ressources $\{R_1,\ldots,R_n\}$ sont qualifiés de *ressources d'entrée* de la fonction F si elles sont utilisées par F pour s'accomplir. Soit $\mathbb{R} = \{R_1,\ldots,R_n\}$ l'ensemble de ressources d'entrée de la fonction F. Nous pouvons distinguer deux cas :
– $\mathbb{R}$ est un ensemble non-complet. Nous avons :

$$fm_{\,1}(R_1) \vee \cdots \vee fm_{\,n}(R_n) \rightarrow \neg ok(F) \tag{2.4.3}$$

ou on déduit

$$ok(F) \rightarrow \neg fm_{\,1}(R_1) \wedge \cdots \wedge \neg fm_{\,n}(R_n) \tag{2.4.4}$$

où fm_i est un des modes de défaut spécifique de R_i.
L'expression 2.4.4 permet de distinguer la différence entre la relation *ressources d'entrée / fonction* avec les relations *sous-items / item-parent* qui sont exprimés par la proposition 2.4.1.
Dans la proposition 2.4.1, si un *item-parent* est en mode *ok*, nous pouvons conclure que ses *sous-items* sont en mode *ok*. La proposition 2.4.4 exprime le fait que si une fonction est en mode *ok*, il n'est pas évident que ses *ressources d'entrée* soient en mode *ok*. En effet, chaque ressource R_i peut voir plusieurs rôles qui supportent le fonctionnement de différentes fonctions. Supposons que l'ensemble des fonctions soit $\mathbb{F}$. Un mode de défaut spécifique $mode(R_i)$ de R_i ne cause pas toujours de défaillance au niveau des fonctions supportées $F_j \in \mathbb{F}$. Revenons à l'exemple de la voiture : la *batterie* est une sous-ressource de deux fonctions déjà évoquées *allumer la lumière* et *démarrer la voiture*. La tension de la *batterie* peut assurer le fonctionnement de la fonction *allumer la lumière* mais elle n'est pas suffisante pour *démarrer la voiture*.
Dans le cadre du modèle FIS, il est possible de préciser la relation (2.4.4) dans le cadre du modèle dysfonctionnel.
– $\mathbb{R}$ est un ensemble complet
Pour un ensemble complet des sous-ressources $\mathbb{R}$ d'une fonction F, si (2.4.4) est vérifié et si toutes les sous-ressources $\mathbb{R}$ sont en mode normal, alors nécessairement la fonction F l'est aussi :

$$ok\,(R_1) \wedge \cdots \wedge ok\,(R_n) \rightarrow ok(F) \tag{2.4.5}$$

Si aucun mode de défaut spécifique n'est défini pour les ressources $\mathbb{R} = \{R_1,\ldots,R_n\}$, 2.4.4 et 2.4.5 devient respectivement 2.4.6 et 2.4.7

$$ok(F) \rightarrow ok\,(R_1) \wedge \cdots \wedge ok\,(R_n) \tag{2.4.6}$$

$$ok(F) \leftrightarrow ok\,(R_1) \wedge \cdots \wedge ok\,(R_n) \tag{2.4.7}$$

– Ressources d'entrée dans FIS dont les modes sont des symptômes dans le module d'analyse diagnostique
Les propositions induisent des m-propositions de symptômes (2.4.6 et 2.4.7 devient respectivement 2.4.8 et 2.4.9)
Par défaut, on écrit

$$ok(F) \rightarrow \perp_{F_1}(R_1) \wedge \cdots \wedge \perp_{F_n}(R_n) \tag{2.4.8}$$

$$ok(F) \leftrightarrow \perp_{F_1}(R_1) \wedge \cdots \wedge \perp_{F_n}(R_n) \tag{2.4.9}$$

F_i est la fonction où R_i est conforme. Autrement dit, R_i est produit par F_i. Ceci sera détaillé ensuite. Il peut exister des OUs et des ETs logiques entre les modes des ressources d'entrée d'une fonction, par exemple $ok(F) \rightarrow \perp_{F_1}(R_1) \wedge (\perp_{F_2}(R_2) \vee \perp_{F_3}(R_3)) \wedge \cdots \wedge \perp_{F_n}(R_n)$. Ce sont des informations complémentaires qui peuvent être données par l'expert. Si aucune information complémentaire n'est donnée par l'expert, 2.4.6 peut être déduit automatiquement.

3. Relations entre fonction et ressources de sortie (lien (3) sur la figure 2.17)

Les ressources de sortie sont définies comme *des sortants* dans FIS. Nous voyons que les liens (3) sur la figure 2.17 peuvent être distingués en *produit* (une fonction produit une ressource) ou *agit* (une fonction agit sur une ressource).

– Un lien de type *produit* $F \rightarrow \mathbb{R}$ peut être exprimé en terme des modes de comportement dans FIS : si une fonction F est en mode de bon fonctionnement, l'ensemble de ses ressources $\mathbb{R}$ produites par F le sont aussi. Dans le module d'analyse diagnostique, il s'agit des symtômes qui sont conformes aux états attendus de la fonction F, noté par $\perp_F$. Ceci est donné par

$$ok(F) \rightarrow \perp_F(R_1) \wedge \cdots \wedge \perp_F(R_n) \tag{2.4.10}$$

Par exemple *lumière* est une ressource de sortie de type observé de la fonction *allumer la lumière*, nous avons *ok (allumer la lumière)*$\rightarrow \perp_F$*(lumière)*.

Notons que pour un lien de type *produire*, les ressources de sortie d'une fonction F sont toujours les ressources dont les modes (dans FIS) sont des symptômes dans l'analyse diagnostique. Une fonction F peut *produire* plusieurs ressources mais une ressource *est produit* par une seule fonction. Si un mode d'une ressource est dit conforme (ou non-conforme), il est sous-entendu que le symptôme est conforme (ou non-conforme) à la fonction qui produit cette ressource.

– Un lien de type *agit* $F \rightarrow \mathbb{R}$ ne peut pas être transformé automatiquement sous forme de proposition logique. Par exemple, la fonction *alimenter en énergie électrique* agit sur le *moteur électrique* est exprimée par un modèle FIS : *(alimenter en énergie*

électrique) → *(moteur électrique)*. Dans ce cas, aucune relation logique ne peut être déduite automatiquement à partir du mode de *alimenter en énergie électrique* et le mode (ou état) de *moteur électrique*. En effet, nous n'avons pas *ok (alimenter en énergie électrique)* → *ok (moteur électrique)* et pas nécessairement *ok (moteur électrique)*→ *ok (alimenter en énergie électrique)*. Dans ce cas, il faut que l'expert intervienne pour compléter le lien de type *agir*. Par exemple *haute-tension(alimenter en énergie électrique)*→ *grillé(moteur électrique)*.

Notons que les ressources qui sont liées avec les liens de type *agir* dans FIS sont toujours des ressources dont les modes sont des d-modes dans le modèle transformé pour l'analyse diagnostique. La nature des liens *fonction / ressource de sortie* est résumée sur la figure 2.18.

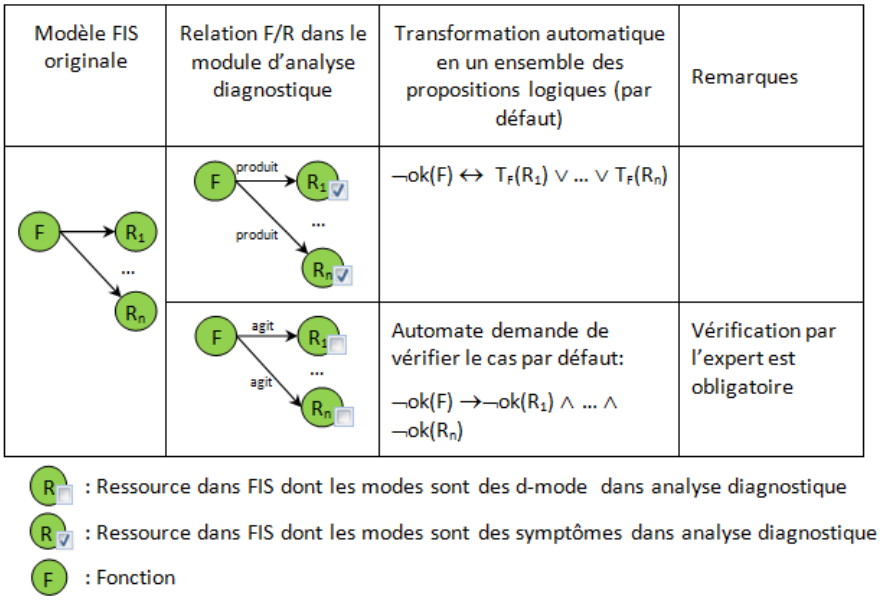


FIG. 2.18 – La nature des liens *fonction/ ressources de sortie*

Graphiquement, les relations *ressources d'entrée / fonction* et *fonction/ressources de sortie* représentées dans FIS originale est transformé en un graphe dans le module d'analyse diagnostique, dont

- chaque noeud est soit une fonction soit une ressource (on l'appelle de façon générique item). Chaque ressource est associée à *une case à cocher*.
- chaque arc orienté représente un lien entre la ressource d'entrée d'une fonction ou les ressources de sortie d'une fonction (figure 2.19). L'arc normal (qui est distingué de l'arc en pointillé) est utilisé pour exprimer l'hypothèse que l'ensemble des ressources d'entrée d'une fonction est toujours complet et qu'une ressource de sortie est produite par une seule fonction. Afin d'aboutir à une transformation automatique de FIS, chaque lien *fonction / ressource de sortie* doit être précisé. Il est soit un lien *produit* soit un lien *agit*.

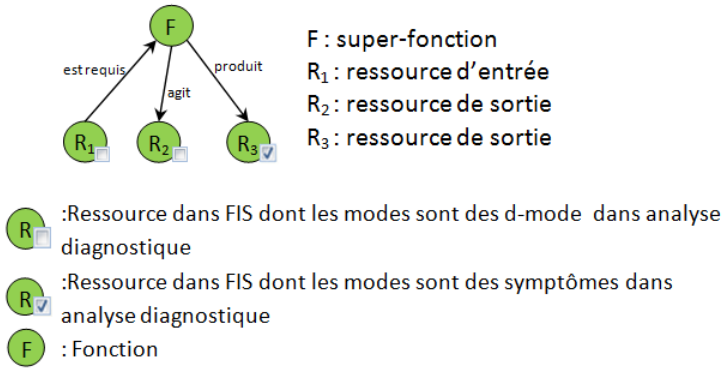


FIG. 2.19 – Représentation d'une relation Sous-ressources/Fonction

Jusqu'ici, nous avons analysé les types de relations *fonction/ressource* dans FIS en faisant le lien avec les modes et le modèle hiérarchique que nous avons reformulé dans la section 2.2. Toutes ces relations sont résumées sur le tableau de la figure 2.20.

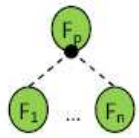
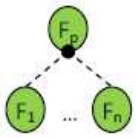
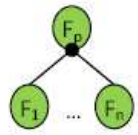
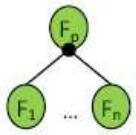
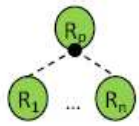
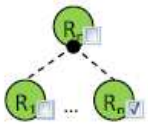
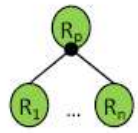
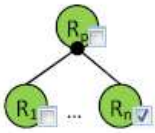
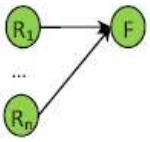
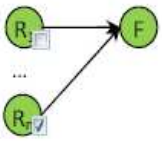
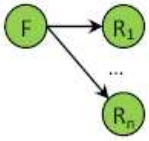
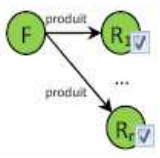
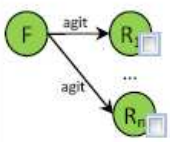
2.4.3 Graphe Fonctions/Ressources (F/R)

Les relations résumées sur le tableau de la figure 2.20 peuvent être représentées sur un graphe F/R (fonction/ ressource) globale, comme sur la figure 2.21. Comme évoqué dans la section 2.4.2, la nature des ressources liées avec des liens *produit* et *agit* peut être déterminée automatiquement : les ressources liées aux liens de type *produire* sont toujours les ressources dont les modes FIS sont des symptômes dans l'analyse diagnostique ; les ressources qui sont liées avec des liens de type *agir* dans FIS sont toujours des ressources dont les modes dans FIS sont des d-modes dans le modèle transformé pour l'analyse diagnostique. Les *cases à cocher* (représentant des informations complémentaires de l'expert sur les types de ressources) n'apparaissent donc pas pour ces ressources sur le graphe F/R. Ceci permet d'éviter les erreurs de l'expert dans la modélisation. Par exemple, l'expert ne coche pas une ressource liée par un lien *produit* car les modes FIS de ces ressources sont des d-modes dans l'analyse diagnostique.

La figure 2.21 peut être représentée plus simplement par la figure 2.22 en ignorant les boîtes qui représentent les systèmes et les sous-systèmes.

A partir du graphe F/R déduit et enrichi à partir du modèle FIS, dans ce module d'analyse diagnostique, le graphe F/R sera traduit en un ensemble de proposition logique qui peut être exploité dans le calcul de diagnostic. Cette transformation est résumée sur la figure 2.20.

Les relations entre les items dans FIS, qui sont transformées sous forme de propositions logiques dans le module d'analyse diagnostique, sont exploitées pour propager des modes de défauts ou des défaillances à partir des observations pour localiser les défauts. Dans le problème d'analyse diagnostique à base de modèle de consistance, nous pouvons exploiter ces relations avec les modèles de comportement disponibles (implicites ou explicites) pour les

	Cas	Modèle FIS originale	Relation F/R dans le module d'analyse diagnostique	Transformation automatique en un ensemble des propositions logiques (transformation par défaut)	Remarques	
1	Abstraction partielle Fonction/Sous-fonctions			$\neg ok(F_1) \vee \neg ok(F_n) \rightarrow \neg ok(F_p)$		Les modes de défaut (ou défaillance) spécifiques peuvent être introduits. Une extention en forme de m-propostion (forme conjonctive normal ou forme disjonctive normal) peut être fait par l'expert, par exemple $fault_1(F) \leftrightarrow (fault_2(R_i) \vee fault_3(R_j)) \wedge fault_4(R_k)$
2	Abstraction complète Fonction/Sous-fonctions			$\neg ok(F_1) \vee \neg ok(F_n) \leftrightarrow \neg ok(F_p)$		
3	Abstraction partielle Ressource/Sous-ressources			$\neg ok(R_1) \vee T(R_n) \rightarrow \neg ok(R_p)$		
4	Abstraction complète Ressource/Sous-ressources			$\neg ok(R_1) \vee T(R_n) \leftrightarrow \neg ok(R_p)$		
5	Ressources d'entrée/ Fonction			$\neg ok(R_1) \vee T(R_n) \leftrightarrow \neg ok(F)$	$\{R_1,...R_n\}$ est un ensemble complet (le cas considéré dans ce travail). Transformation par défaut est juste quand il n'y a pas le OU logique entre les flèches.	
6	Fonction /Ressources de sortie			$T_F(R_1) \vee ... \vee T_F(R_n) \leftrightarrow \neg ok(F)$		
				Automate demande de vérifier le cas par défaut: $\neg ok(F) \rightarrow \neg ok(R_1) \wedge ... \wedge \neg ok(R_n)$	Vérification par l'expert est obligatoire	

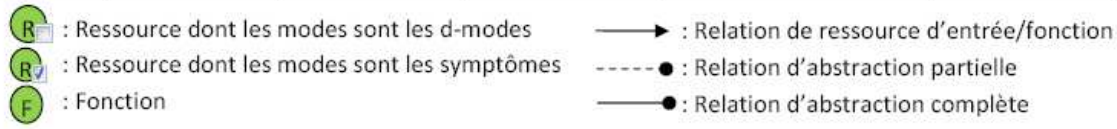


FIG. 2.20 – Transformation de FIS en un ensemble des propositions logiques pour l'analyse diagnostique

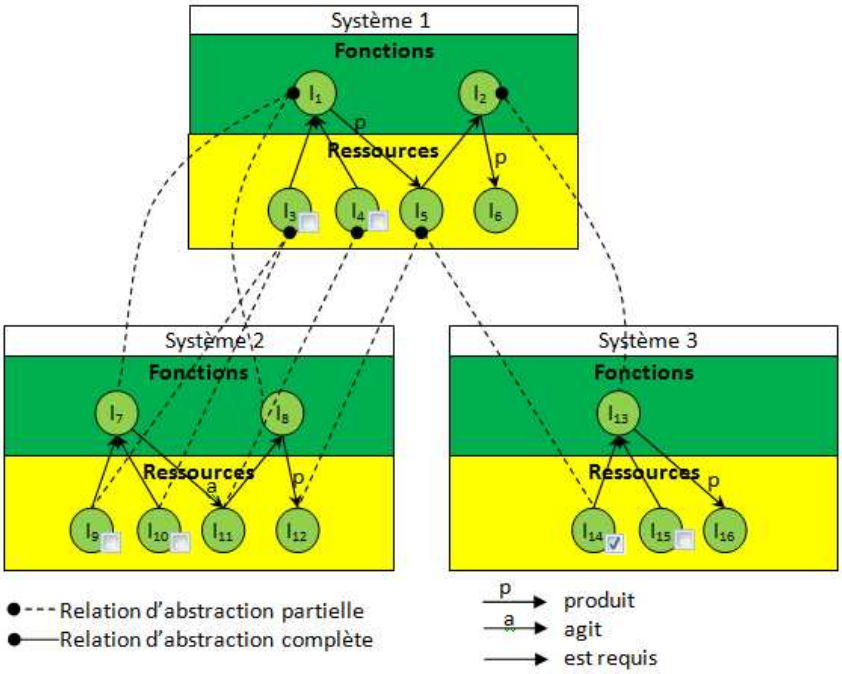


FIG. 2.21 – Un exemple du graphe F/R déduit de la modélisation FIS pour l'analyse diagnostique

items pour chercher des diagnostics. Comme le modèle comportemental peut être disponible seulement pour certaines ressources, les tests peuvent être réalisés au niveau d'abstraction des fonctions et des ressources. Dans les explications des symptômes (section 2.1.5, page 33), les modes des fonctions ou des ressources sont remplacés grâce aux relations F/R. Ceci permet de conduire à des diagnostics au niveau des ressources. La localisation des défauts au niveau des ressources est le but final de l'analyse diagnostique de panne pour les systèmes industriels. Les relations de support peuvent être exprimées par des liens entre les items via des variables.

Cohérence du graphe F/R

La décomposition hiérarchique des fonctions et des ressources forment deux hiérarchies en parallèle. Il faut assurer la cohérence entre elles. Cette cohérence est exprimée par la propriété suivante :

Propriété 2. Soient $\mathbb{R}$ et $\mathbb{R}'$ respectivement les ensembles des ressources (les ressources dont les modes sont les *d-modes* dans l'analyse diagnostique) liées aux fonctions F et F' dans les relations de type *est requis* et *agit* (le lien *produit* n'est pas considéré), si F est sous-fonction de F' dans une relation d'abstraction, alors $\mathbb{R}$ doivent être sous-ressources de $\mathbb{R}'$ dans une relation d'abstraction.

Prenons l'exemple des items sur le graphe F/R de la figure 2.22 (la figure 2.22 est sim-

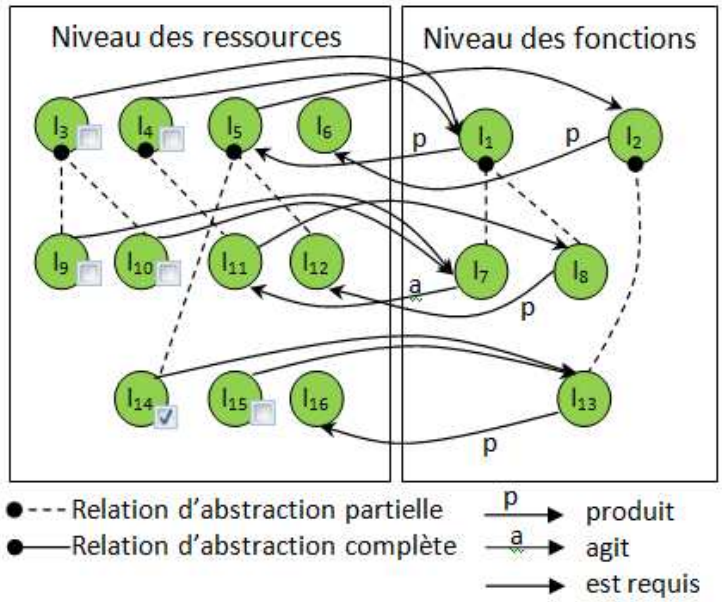


FIG. 2.22 – Graphe F/R (Fonctions/Ressources)

plification de la figure 2.21 obtenue en ignorant les boîtes qui représentent les systèmes et les sous-systèmes). $\{I_9, I_{10}, I_{11}\}$ sont les ressources liées (est requis/agit) à la fontion I_7. $\{I_3, I_4, I_5\}$ sont les ressources liées (est requis/agit) à la fontion I_1. Comme I_7 est une sous-fonction de I_1, il est raisonnable que $\{I_9, I_{10}, I_{11}\}$ soit l'ensemble des sous items de $\{I_3, I_4, I_5\}$ (I_9 et I_{10} sont les sous-items de I_3. I_{11} est le sous-item de I_4).

Dans la procédure de construction du modèle hiérarchique, en se basant sur la propriété (2), l'automate peut vérifier la cohérence du graphe F/R et signaler à l'expert des erreurs de modélisation ou proposer à l'expert de compléter le graphe. Par exemple sur le graphe présenté de la figure 2.22, $\{I_{14}, I_{15}, I_{16}\}$ sont les ressources liées à la fontion I_{13}, $\{I_5, I_6\}$ sont les ressources liées à la fontion I_2. Comme I_{13} est une sous-fonction de I_2, alors $\{I_{14}, I_{15}, I_{16}\}$ doivent être les sous-ressources de $\{I_5, I_6\}$. Ces liens entre $\{I_{15}, I_{16}\}$ et $\{I_5, I_6\}$ ne sont pas représentés sur le graphe de la figure 2.22. Ceci est une erreur de modélisation de l'expert. L'automate va demander à l'expert de vérifier la modélisation entre $\{I_{15}, I_{16}\}$ et $\{I_5, I_6\}$.

Dans l'analyse diagnostique d'un système, nous nous intéressons aux défauts au niveau des ressources. Il faut donc remplacer chaque fonction par ses sous-ressources. Ce remplacement s'appuie à la fois sur les deux hiérarchies parallèles des fonctions et des ressources et sur les relations sous-ressources/fonction-parent.

2.4.4 Modélisation interactive par l'expert en utilisant la modélisation structuro-fonctionnelle pour l'analyse diagnostique

Jusqu'ici, nous avons formulé les différents éléments d'un problème de diagnostic interactif en se basant sur le modèle FIS. Il reste encore à examiner comment ces entités seront construites au fur et à mesure des interactions. Dans la figure 2.23, nous voyons que la tâche de construction du modèle du système est réalisée par l'expert. Elle consiste à expliciter les connaissances recueillies. La construction du modèle peut être divisée en deux étapes : *la modélisation FIS* et *la modélisation complémentaire pour le module d'analyse diagnostique* (figure 2.23). *La modélisation FIS* donne des informations globales qui peuvent être exploitées pour les modules intégrés dans FIS (analyse de risques, analyse diagnostique...). *La modélisation complémentaire* présentée dans ce travail est nécessaire pour l'analyse diagnostique s'appuyant sur les données recueillies dans FIS.

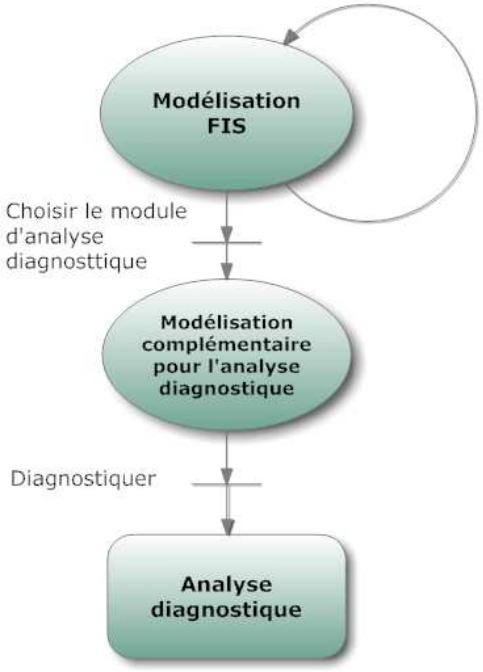


FIG. 2.23 – Modélisation pour l'analyse diagnostique en s'appuyant sur FIS

Modélisation FIS

Au fur et à mesure, le modèle FIS est construit par l'expert à travers les questions suivantes :

1. Quelles sont les fonctions globales du système ? La procédure de diagnostic commence quand un dysfonctionnement du système est constaté. Normalement, l'expert commence en testant les fonctions globales du système, puis il affine en les décomposant.

2. Pour chaque fonction F_i,
   – quelles sont les ressources d'entrée et les ressources de sortie de F_i ?
   – quelles sont les sous-fonctions de F_i ?
3. Pour chaque ressource R_i,
   – quelles sont les fonctions qui associées à R_i ou bien les fonctions dont R_i jouent le rôle de ressource d'entrée ? Une ressource peut elle supporter le fonctionnement de plusieurs fonctions ?
   – quelles sont les sous-ressources de R_i ?
4. Les modèles comportementaux associés aux modes des items peuvent-ils être renseignés ? L'expert peut décrire des modèles comportementaux au fur et à mesure de la procédure de diagnostic.
5. Existe-il des connexions (partielles/complètes) entre les modèles élémentaires au niveau fonctionnel et au niveau des ressources ?

Nous voyons que les approches détaillées dans (Flaus [2008]) dirigent la procédure de décomposition interactive. En effet, les points (2) et (3) montrent une possibilité de décomposer structurellement un système et d'aboutir à un niveau plus fin.

Modélisation complémentaire pour le module d'analyse diagnostique

Quand nous passons au module d'analyse diagnostique, deux types d'informations complémentaires sont nécessaires :

– Pour chaque ressource, quelle est la nature des modes associés avec les ressources quand nous passons au module d'analyse diagnostique : *d-modes* ou symptôme (section 2.4.1) ?
– Quelle est la nature de chaque lien *fonction / ressource de sortie* : *agir* (une fonction agit sur une ressource) ou *produire* (une fonction produit une ressource) (section 2.4.2).

Pour compléter ces informations, quand nous passons au module d'analyse diagnostique, une interaction homme-automate se manifeste de la façon suivante :

1. Déterminer les types de lien *fonction/ressource*

   L'automate demande à l'expert de préciser les liens *fonction / ressource de sortie* : chaque lien *fonction / ressources de sortie* sera étiqueté soit par *agit* soit par *produit*.
2. Déterminer la nature des ressources pour le module de diagnostic

   Pour une ressource qui est liée au lien *produit*, les modes sont des symptômes (conforme / non-conforme).

   Pour une ressource qui est liée au lien *agit*, les modes sont des d-modes (ok / modes de défaut spécifiques / cfm).

   Pour les ressources liées aux liens *est requis*, une case à cocher est ajoutée automatiquement. Cette case à cocher permet à l'expert de déterminer la nature de la ressource

dans le module de diagnostic. Si la case est cochée, les modes de la ressource sont des symptômes (conforme/non-conforme). Dans le cas contraire, les modes de la ressource sont des d-mode (ok /cfm).

Notons que si une ressource est liée à la fois à un lien *est requis* et à un autre lien *produit* (ou *agit*), la case à cocher de la ressource n'apparaît pas parce qu'il est déjà précisé automatiquement par l'automate selon le lien *produit* (ou *agit*).

3. Compléter les modes par défaut pour les ressources

   Un symptôme {conforme, non-conforme} est ajouté automatiquement pour chaque ressource qui est liée au lien *produit*.

   Un d-mode {ok, cfm} est ajouté automatiquement pour les ressource du lien *agit* mais aussi les fonctions.

   Pour les autres ressources : si la case est cochée, un symptôme {conforme, non-conforme} est ajouté automatiquement. Si la case n'est pas cochée, {ok, cfm} est ajouté automatiquement pour la ressource.

Un exemple simple du passage du modèle FIS original au module de diagnostic est présenté sur la figure (2.24)

Dans ce module d'analyse diagnostique, la description comportementale peut être intégrée dans FIS. La description comportementale consiste à enrichir le modèle structuro-fonctionnel par le modèle comportemental. La composition du modèle comportemental peut être partielle ou complète. Cela veut dire que le modèle comportemental est disponible pour certains items dans le système. Les autres sont non-modélisés. Les modèles comportementaux ajoutés pour les items permettent de construire le modèle de référence pour la tâche de détection des symptômes. Ceci sera détaillé plus tard dans le chapitre 4. Dans le chapitre 3, nous ne nous intéressons pas au modèle comportemental.

2.4.5 Conclusion

Dans cette partie, nous avons rappelé et détaillé FIS comme un outil pour décrire le modèle de systèmes à différents niveaux de décomposition. Comme FIS est conçu initialement pour l'analyse des risques, nous avons adapté le modèle issu de FIS pour une exploitation par un module d'analyse diagnostique. Dans ce but, un graphe F/R a été déduit de FIS. Sur ce graphe F/R, nous avons reformulé et détaillé les relations entre items (fonction/ressource) en terme de modes de comportement. En comparaison avec le modèle FIS original, certaines informations complémentaires sont exigées dans ce module d'analyse diagnostique. Ceci permet de faire le lien entre le modèle comportemental, le modèle d'abstraction pour l'analyse diagnostique reformulé dans (2.2) et le modèle structuro-fonctionnel FIS. Cette formulation est la base pour étudier les problèmes que nous présentons dans les chapitres 3 et 4.

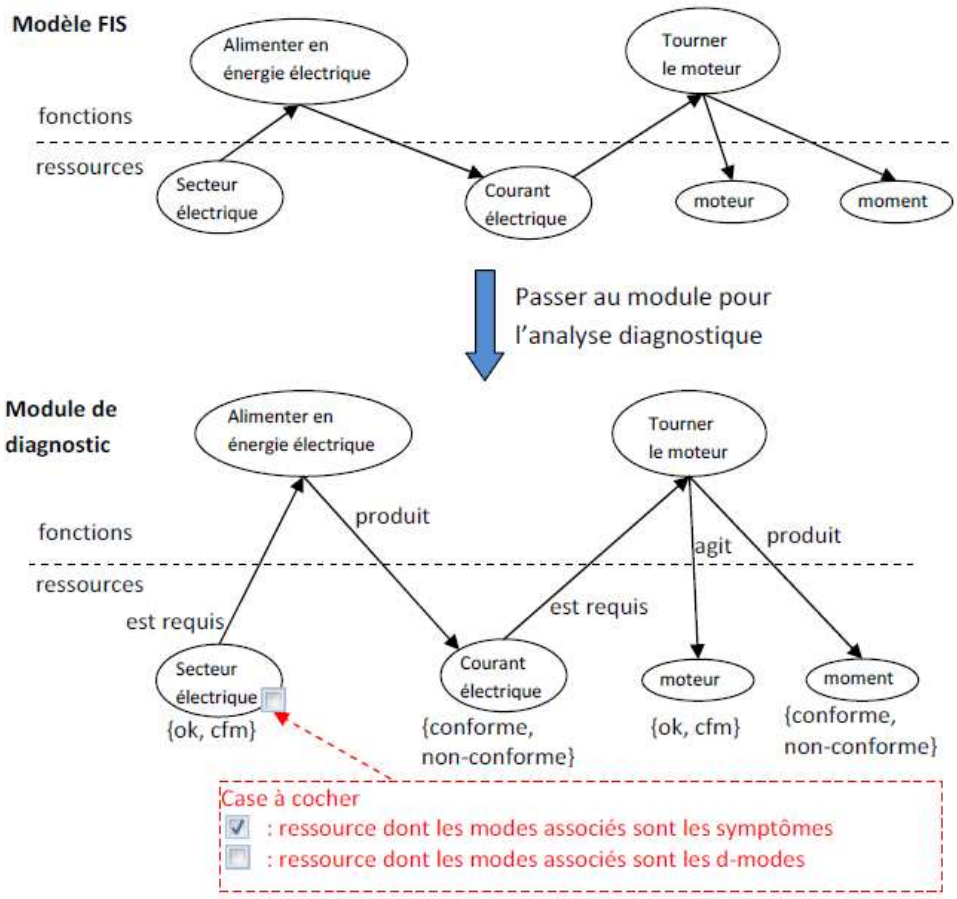


FIG. 2.24 – Exemple sur la modélisation complémentaire pour le module d'analyse diagnostique en s'appyant sur FIS

2.5 Conclusion

Ce chapitre est consacré à la formulation itérative et interactive de modèles structuro-fonctionnels du système. Nous avons rappelé tout d'abord les éléments formulés par l'approche hybride de diagnostic DX-FDI. Nous avons développé ensuite en détail certains éléments en visant à la construction d'un modèle d'abstraction. En se basant sur cette formulation, nous avons défini un modèle hiérarchique via des éléments de base du système : variable, modèle élémentaire et item (fonction ou ressource). Ce sont les éléments que l'expert utilise pour alimenter au fur et à mesure le modèle du système.

Dans une deuxième phase, le modèle structuro-fonctionnel FIS est rappelé de façon à pouvoir exploiter les fonctions dans l'analyse diagnostique car le modèle fonctionnel permet de décrire le système plus facilement et plus simplement. Quand la notion de fonction est introduite, en plus des relations hiérarchiques sous-item/super-item, nous avons dans FIS des relations fonction/ressource. Jusqu'ici, la contribution principale de notre travail est de rapprocher les éléments de formulation pour un problème de diagnostic du modèle FIS. Ce rapprochement

a permis d'aboutir à un module d'analyse diagnostique en s'appuyant sur FIS. Ceci donne un outil pour résoudre les problèmes de diagnostic itératif évoqués dans l'introduction de ce manuscrit. Cet outil permet de décrire le modèle du système au fur et à mesure dans le processus de diagnostic sans exiger beaucoup d'information sur le système.

Ce chapitre donne une vue globale sur la méthode de modélisation que nous utilisons ensuite pour résoudre les deux premiers problèmes présentés ensuite dans les chapitres 3 et 4. Le modèle fonctionnel n'est pas exploité dans le chapitre 5.

Chapitre 3

Diagnostic itératif basé sur le modèle structuro-fonctionnel FIS

3.1 Introduction

Dans la communauté du MBD (Model Based Diagnosis), la plupart des approches proposées requièrent un modèle comportemental détaillé et complet du système à diagnostiquer. Dans certains cas, ce pré-requis est difficile à satisfaire. Considérons l'exemple du problème de diagnostic d'une voiture en panne qu'un garagiste réalise souvent dans son métier, ou le travail d'un informaticien pour dépanner un système informatique qui comporte des milliers de postes de travail ou le problème de diagnostic des dysfonctionnements dans un système organisationnel. Dans ces problèmes, les dysfonctionnements peuvent être causés par les ressources (les ressources au sens large sont des ressources matérielles, humaines, informationnelles, etc.). Il apparaît aussi que la plupart des approches qui s'appuient sur MBD (Model Based Diagnosis) conviennent mal aux travaux des services de maintenance parce que toutes ces approches partent toujours de l'hypothèse qu'il existe un modèle explicite au départ. En réalité, le modèle n'est souvent découvert que lorsqu'un défaut survient et qu'il faut dépanner le système et, de surcroît, rarement explicité. Le fonctionnement du système est décortiqué petit à petit et un modèle se construit. Nous souhaitons donc présenter dans ce travail une approche où l'analyse diagnostique est considérée comme un processus itératif.

Le modèle du système sera découvert et formulé à l'aide de l'expertise se situant à différents niveaux de détail. Par exemple, une voiture peut être décomposée en sous-systèmes : système d'injection, système d'allumage, système électrique... Chaque sous-système peut être décomposé en d'autres sous-systèmes. Par exemple, le système électrique peut être décomposé en un système d'éclairage et en un système de démarrage électrique. Au cours de la description du système, seules certaines parties du système seront détaillées selon les symptômes collectés par l'expert. Il ne devrait donc pas être nécessaire de décrire complètement un système en une

étape avant l'analyse diagnostique.

Par ailleurs, pour ce qui concerne la description du comportement de chaque élément, on aboutit rapidement à un modèle de comportement très complexe qui doit être décrit par des relations de natures très différentes. Ces relations peuvent être des équations mathématiques statiques ou dynamiques, des relations continues ou discrètes. Pour aider un service de maintenance, il faut chercher une méthode de diagnostic qui permette d'exploiter au mieux la connaissance de l'expert dans la localisation des défauts sans avoir besoin nécessairement de décrire toutes ces relations, mais qui permette de décrire des relations simples entre les éléments du système et qui permette de les décomposer au fur et à mesure.

Par conséquent, le problème que nous posons est exprimé par les trois points principaux suivants :

- dans certains cas, les approches existant dans la littérature, qui s'appuient sur MBD et qui exigent un modèle de comportement complet et détaillé du système, conviennent mal au travail des services de maintenance.
- en cas de dysfonctionnement, l'analyse diagnostique est un processus où le fonctionnement du système sera décrit au fur et à mesure avec différents niveaux de détail.
- le modèle d'un système pouvant devenir rapidement complexe, il est nécessaire de mettre au point un outil de diagnostic qui permette d'exploiter au mieux la connaissance de l'expert sans requérir beaucoup d'explicitation de connaissance.

Ce problème invite à étudier un nouvelle approche de diagnostic avec des interactions durant la phase de test. L'analyse diagnostique est présentée comme un processus qui comporte plusieurs étapes itératives (figure 3.1). Chaque étape est initiée à une interaction home-automate. A chaque étape, nous supposons implicitement que l'expert peut :

- effectuer un test grâce à son expertise pour améliorer la localisation des défauts sachant un ensemble de défauts possibles, calculés par l'automate.
- expliciter sa connaissance sur la partie du système qui est testée sous forme d'interconnexion et la formulation de symptômes : le résultat du test est conforme aux attentes (modèles de référence parfois implicites) ou non.

Dans ce processus, l'expert s'appuie souvent sur son intuition basée sur son expérience pour localiser les défauts. Pourtant, quand le système est complexe, l'expert rencontrera des difficultés dans la capitalisation des informations collectées (modèle du système, symptômes...) pour chercher toutes les explications possibles. Dans ce chapitre, nous proposons un outil d'aide au diagnostic pour accompagner l'expert durant ce type de processus de diagnostic. Les trois étapes globales d'un problème de diagnostic présentées sur la figure 1.9 (page 1.9) sont maintenant détaillées sur la figure 3.1.

Dans ce chapitre, nous allons donner tout d'abord le principe de l'approche dans la section 3.2. Puis nous détaillons dans la partie 3.3 la procédure d'intéraction et l'algorithme permettant la résolution du problème de diagnostic à chaque itération (blocs (1) et (2) sur la figure 3.1).

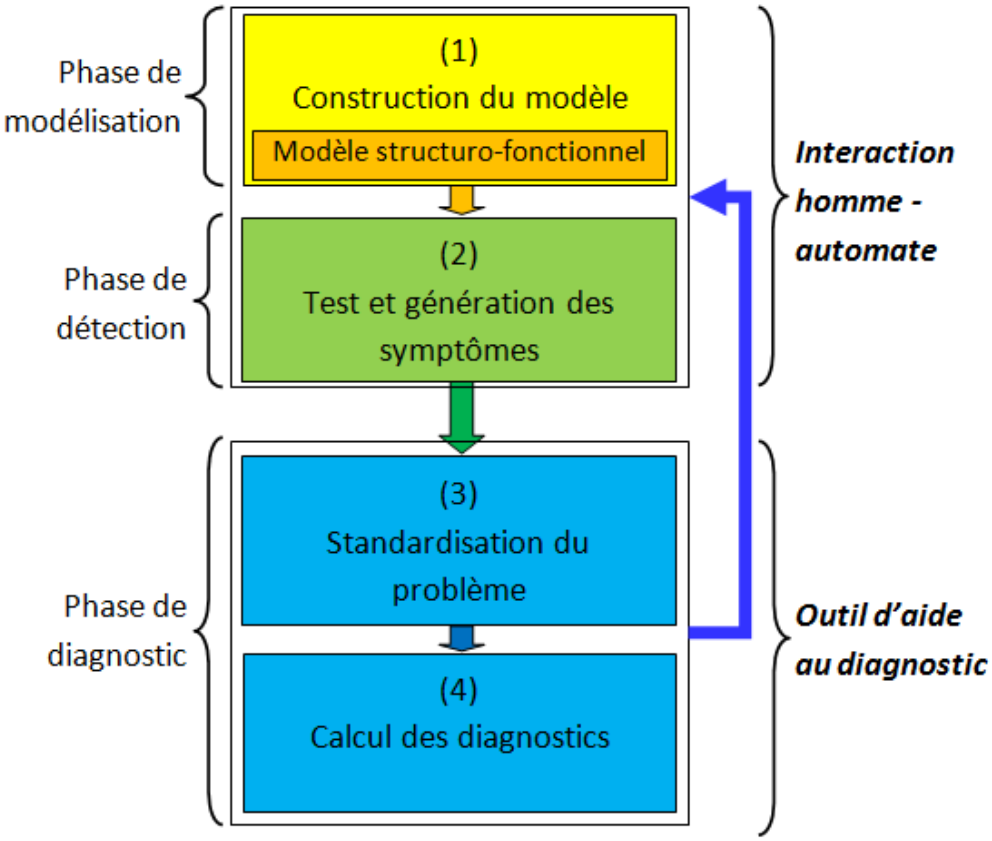


FIG. 3.1 – Diagnostic itératif basé sur le modèle structuro fonctionnel FIS

Ensuite, un exemple d'application et un outil d'aide au diagnostic sont présenté dans 3.4.

3.2 Principe général

Dans cette partie, nous allons résumer la méthode de modélisation utilisée dans ce problème qui permet la capitalisation des connaissances de l'expert lors de la formulation d'un problème de diagnostic à une itération du processus. Puis, nous présentons de façon générale le principe de résolution. Cette démarche reprend les idées de (Flaus et Pham [2010]), approche qui a été utilisée pour les systèmes organisationnels (Flaus *et al.* [2010]),(Karagiannis [2010]) et vise à automatiser la construction des modèles dysfonctionnels à partir du graphe F/R.

3.2.1 Méthode de modélisation

La méthode de modélisation proposée dans cette partie est la méthode de modélisation basée sur FIS présentée dans le chapitre 2. Elle est intégrée dans le module de diagnostic du

logiciel X-Risk (section 2.4). Cette méthode permet de modéliser le système par des fonctions et de considérer différents types de ressources : ressources matérielles, humaines, informationnelles, etc. Cette approche offre un cadre pour l'analyse diagnostique pour variés types de systèmes : système organisationnel, système industriel, etc.

Capitalisation des connaissances de l'expert pour la formulation d'un problème de diagnostic

Cette phase de modélisation est représentée par les blocs (1) et (2) sur la figure 3.1. Notons que pour la modélisation dans ce chapitre, nous ne nous intéressons pas au modèle comportemental. Le modèle comportemental sera introduit dans le chapitre 4. En résumé, à chaque interaction du processus de diagnostic, le problème formulé par l'expert comporte :

1. le modèle structuro-fonctionnel construit par l'expert. Dans ce modèle, les informations disponibles sont :
   - une liste d'items (fonction, ressource) avec des modes de défaut spécifiques associés (des modes de défaillances/défauts de chaque fonction et ressource). Notons que les deux modes {ok, cfm} sont ajoutés automatiquement pour chaque item quand il est ajouté dans le système. Le plus souvent, chaque item a seulement deux modes {ok, cfm}.
   - des fonctions et des sous-fonctions organisées de façon hiérarchique (chaque fonction a une liste de fonctions enfants).
   - des ressources et des sous-ressources organisées elles aussi de façon hiérarchique.
   - des liens entre les fonctions et les ressources (des liens entrant/sortant) décrits par le graphe F/R.
   - l'abstraction des modes de défaillances, qui donne pour chaque mode de défaillance/défaut d'une fonction ou ressource, le mode de défaillance ou défaut de la fonction ou ressource parent.
   - un modèle de dysfonctionnement exprimant des relations logiques entre les modes de défaillances des différents éléments. Ces relations entre les modes des items peuvent être déduites par défaut à partir du graphe structuro-fonctionnel. Des informations complémentaires peuvent être ajoutées pour décrire ces relations. Elles sont exprimées par des propositions logiques. Par exemple, $fm\ (I_1) \rightarrow fm\ (I), fm\ (I_1) \wedge fm\ (I_2) \rightarrow fm\ (I)$, etc.

   Le graphe structuro-fonctionnel peut être transformé et traduit en un ensemble de propositions logiques pour alimenter l'algorithme de diagnostic.
2. des symptômes formulés sous forme de clauses ou de monômes de modes.
3. de façon optionnelle, la probabilité de défaillance des modes de base (c'est à dire ceux sans causes explicitées)

3.2.2 Vues générales sur la résolution du problème

Avant de détailler la méthode pour résoudre le problème de diagnostic formulé à chaque interaction dans une procédure de diagnostic, nous exprimons ici la liste des spécifications que notre approche doit permettre de résoudre :

- elle doit permettre de prendre en compte des défauts multiples. Cela veut dire que deux ou plus de deux défauts peuvent se manifester en même temps dans le système.
- elle doit gérer des relations hiérarchiques existant dans le système. Comme le système est décrit aux différents niveaux d'abstraction, pour éviter des diagnostics non-minimaux, il ne faut pas qu'il existe dans le support d'un même diagnostic le mode de défaut d'un item et celui de son super-item.
- elle doit bien sûr fournir un résultat prenant en compte le graphe de dysfonctionnement du modèle FIS
- elle doit permettre de prendre en compte les tests consistants, point particulièrement important en contexte interactif.
- elle doit fournir un classement des diagnostics afin de guider l'expert durant le processus de diagnostic.

A partir du problème énoncé, formulé à chaque itération du processus de diagnostic et des besoins sur un outil d'aide au diagnostic, nous proposons une méthodologie de résolusion pour calculer les diagnostics à partir du problème énoncé tout en veillant la performance de l'outil résultant. Le bloc (3) et (4) sur la figure 3.1 est détaillé en sous-étapes décrites sur la figure 3.2, page 3.2. Chaque sous-étape est détaillée dans la suite.

3.3 Procédure et algorithme de diagnostic

Dans cette partie, nous allons détailler l'algorithme pour calculer les diagnostics à partir du modèle formulé et les symptômes collectés par l'expert.

3.3.1 Interaction avec l'utilisateur

L'utilisateur commence à modéliser son système (ou s'appuie sur un modèle déjà existant) en s'appuyant sur FIS et formuler un test. Les tests pourront être exprimés de deux façons selon qu'ils sont consistants ou inconsistants :

- Si le test est inconsistant, l'expert fournit le test au système comme une disjonction de modes (ou de symptôme) de défauts de différents items :

$$\neg mode_{i1}(I_1) \vee \neg mode_{j1}(I_1) \vee \cdots \vee \neg mode_{ik}(I_k) \vee \cdots \vee \neg \top(I_n)$$

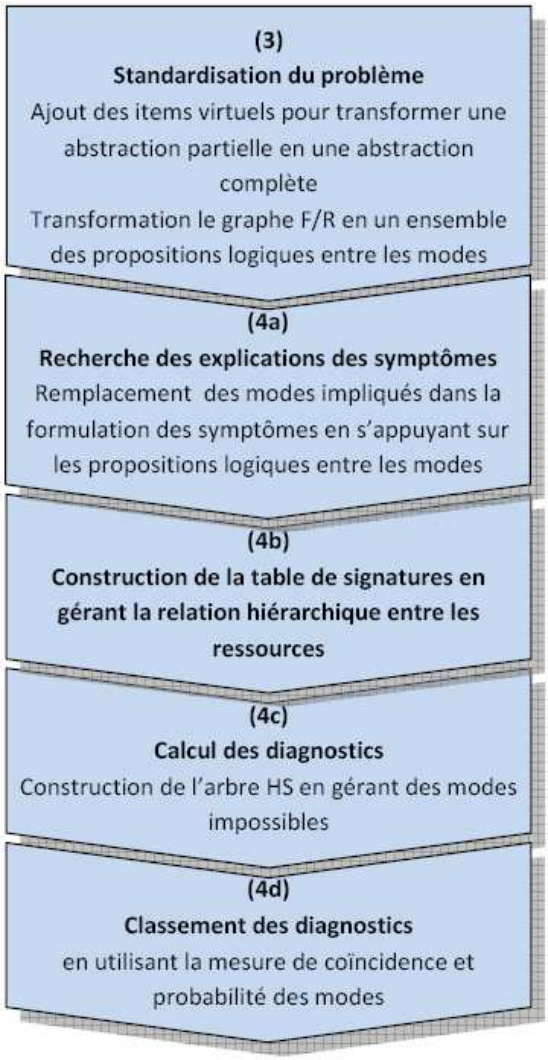


FIG. 3.2 – Calcul des diagnostics

– Si le test est consistant, l'expert fournit le test au système comme une conjonction de modes (ou de symptôme) des items en bon fonctionnement :

$$ok(I_1) \wedge \cdots \wedge ok(I_k) \wedge \cdots \wedge \perp(I_n)$$

Le système fournit alors en utilisant ces observations, une liste de diagnostics, c'est à dire une liste d'ensemble de modes de défaillance permettant d'expliquer les observations, compte tenu du modèle FIS.

Il est possible que cette liste soit vide. Dans ce cas, cela signifie que l'expert a effectué une erreur de modélisation ou une erreur d'observation.

Lorsque la liste des disgnostics n'est pas vide, deux cas de figure se présentent : soit la liste est suffisamment précise pour permettre une intervention sur les éléments concernés soit elle n'est pas suffisamment précise. Dans ce second cas, l'expert va réaliser une itération dans la

démarche de diagnostic,

- soit en ajoutant un nouveau test avec le modèle existant
- soit en détaillant le modèle, c'est à dire en décomposant une ou plusieurs ressources et/ou fonctions en sous items et décrivant la structure FIS associée (liens ressources/fonctions et fonctions/ressources).

A partir de ces nouvelles informations, il relance un calcul des diagnostics et poursuit ainsi jusqu'à obtenir une liste de diagnostics suffisamment précise.

L'approche proposée lui permet de gérer la complexité des informations qu'il accumule au fur et à mesure, à la fois sur le modèle du système et sur les tests réalisés, et de lui fournir une liste des défaillances possibles de son système.

3.3.2 Algorithme de diagnostic

Dans cette partie, nous allons détailler les outils théoriques et les techniques utilisées pour concevoir l'algorithme d'aide au diagnostic.

Standardisation du modèle en ajoutant des items virtuels (bloc (3) sur la figure 3.2)

La première étape consiste à compléter les abstractions partielles par des items virtuels. Cette étape correspond au bloc (3) sur la figure 3.1. Le modèle structuro-fonctionnel donné par l'expert à chaque interaction peut ensuite être transformé complètement en un ensemble de propositions logiques.

Exemple 11. Prenons l'exemple du système modélisé par le graphe F/R représenté sur la figure 3.3. L'item I_4 de ce système n'est pas complétement décomposé. Par conséquent le mode virtuel $IV_1 = I_4 \setminus \{I_7, I_8\}$ est ajouté.

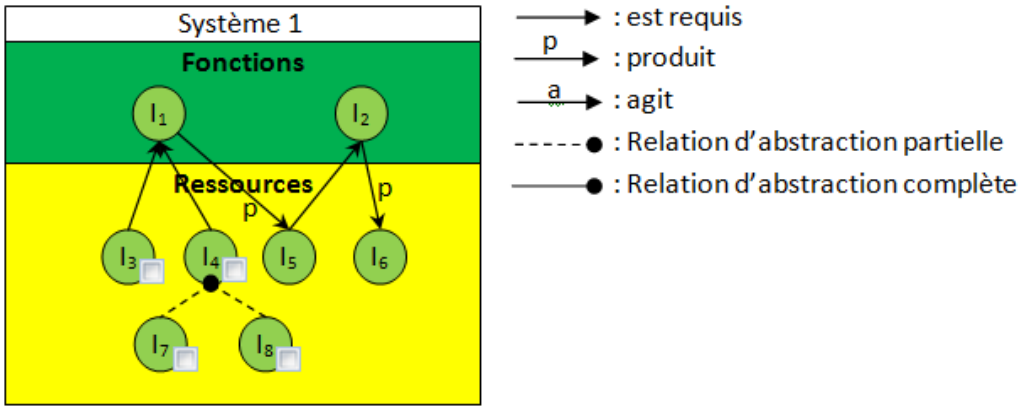


FIG. 3.3 – Modèle FIS d'un système

Par ailleurs les relations logiques suivantes sont automatiquement déduites du graphe F/R selon les règles résumées sur la figure 2.20.

$$\begin{aligned} ok\ (I_6) &\leftrightarrow ok\ (I_2) \\ ok\ (I_2) &\leftrightarrow \bot(I_5) \\ \bot(I_5) &\leftrightarrow ok\ (I_1) \\ ok\ (I_1) &\leftrightarrow ok\ (I_3) \wedge ok\ (I_4) \\ ok\ (I_4) &\leftrightarrow ok\ (I_7) \wedge ok\ (I_8) \wedge ok\ (IV_1) \\ avec\ \ IV_1 &= I_4 \setminus \{I_7, I_8\} \end{aligned} \tag{3.3.1}$$

Recherche des explications des symptômes (bloc (4a) sur la figure 3.2)

La recherche des explications des symptômes consiste à remplacer les modes impliqués dans les symptômes formulés par l'expert. Ce remplacement s'appuie sur l'ensemble de propositions logiques entre les modes transformés du graphe F/R (présenté par le bloc 3). Cette étape de remplacement permet de remplacer tous les modes des fonctions dans les explications des symptômes par les modes des ressources à l'aide des propositions logiques résumées sur la figure 2.20, page 56. Le remplacement des fonctions par des ressources permet de calculer des diagnostics et de localiser des défauts au niveau des ressources qui est le but final du problème de diagnostic de panne.

Les relations d'abstraction ne sont pas exploitées dans cette étape. Elles seront exploitées plus tard de façon à garantir qu'un diagnostic ne comporte pas le mode d'un item-parent et celui de ses items enfant.

De façon générale, ce remplacement est effectué en écrivant les tests sous une forme normale disjonctive en fonction des modes de défaillances des éléments explicatifs.

Exemple 12. Revenons à l'exemple (11) avec le modèle du système représenté sur la figure (3.3). Supposons que l'expert effectue un test et formule un symptôme : $\top(T_1) \rightarrow cfm\ (I_2)$. Une explication au niveau des ressources peut être trouvée en utilisant les propositions (3.3.1) :

$$Expl(\top(T_1)) : cfm\ (I_3) \vee cfm\ (I_4)$$

Gestion de la relation hiérarchique entre les items dans l'explication des symptômes (bloc 4b sur la figure 3.2)

L'analyse diagnostique pour un modèle qui est décrite à différents niveaux de décomposition peut se ramener à un ensemble non-minimal de diagnostics. Cet ensemble est non minimal

parce qu'il existe des diagnostics qui peuvent être expliqués par les autres via des relations hiérarchiques entre les items. De plus, il peut exister dans cet ensemble des diagnostics qui comportent à la fois le mode d'un item parent et le mode d'un de ses items enfants. Ce sont des diagnostics non-minimaux. Un ensemble de diagnostics non-minimal sera difficile à appréhender par l'opérateur. De plus, dans l'analyse diagnostique, il n'est pas nécessaire de détailler au niveau des sous-items qui n'apparaissent pas dans les tests.

Supposons que $\mathscr{T}_n$ est l'ensemble des explications de test (donné par le bloc 4a sur la figure 3.2) obtenu à l'étape n du processus de diagnostic, $Modes(\mathscr{T}_n)$ est l'ensemble de modes impliqués dans $\mathscr{T}_n$. Pour assurer qu'il n'existe pas de relations hiérarchiques entre les modes impliqués dans un diagnostic, nous remplaçons des modes des items parent existant dans $Modes(\mathscr{T}_n)$ de façon récursive jusqu'à ce qu'il n'existe plus de relations hiérarchiques entre les modes dans $Modes(\mathscr{T}_n)$.

Pour remplacer, il nous faut tout d'abord détecter l'existence des relations hiérarchiques. La détection des relations hiérarchiques entre $mode(I_i) \in Modes(\mathscr{T}_n)$ et $mode(I_j) \in Modes(\mathscr{T}_n)$ et le remplacement des modes parent sont réalisés à l'aide du graphe F/R présenté sur la figure 2.22, page 58. En principe, le graphe sur la figure 2.22 montre qu'il existe une relation hiérarchique entre deux items I_i et I_j s'il existe un chemin constitué d'arcs entre eux (des arcs représentant des liens d'abstraction). Un chemin de I_i à I_j permet de dire que I_i est le super-item de I_j. Il est facile de vérifier si un item I_i est le super-item d'un autre item I_j sur le graphe F/R. Nous déterminons tout d'abord les sous-items de I_i dans ce graphe, noté par $SubItems(I_i)$. Si $I_j \in SubItems(I_i)$ alors I_j est le sous-item de I_i. La détection des relations hiérarchiques est réalisée par la vérification de chaque paire de modes $mode(I_i)$ et $mode(I_j)$ dans l'ensemble $Modes(\mathscr{T}_n)$ pour chercher l'existence de chemins entre eux.

$Modes(\mathscr{T}_n)$: **un ensemble de modes impliqués dans** $\mathscr{T}_n$
$Path_{ij}$: **un ensemble d'items sur le chemin de** I_i **à** I_j **(le chemin constitué par des arts représentant des liens d'abstraction) tel que** $I_i \in Path_{ij}$ **et** $I_j \notin Path_{ij}$ (on ne remplace pas les modes de I_j par les modes de ses items-enfant)

Tant que ($\exists mode(I_i) \in Modes(\mathscr{T}_n)$, $\exists mode(I_j) \in Modes(\mathscr{T}_n)$ tel que I_i est le super-item de I_j) **faire**
    [Déterminer $Path_{ij}$]
    Tant que ($\exists mode(I_k)$ tel que $I_k \in Path_{ij}$) **faire**
        [Remplacer $mode(I_k)$ par l'ensemble de modes enfants en ajoutant un item virtuel si la décomposition est partielle]
    Fin Tq
Fin Tq

ALG. 1: Gestion de la relation hiérarchique entre les items

Exemple 13. Supposons que nous ayons un système avec des relations hiérarchiques entre les items qui sont présentés sur la figure 3.4 (abstraction entre les fonctions ou les ressources). Supposons qu'il existe un ensemble des explications $\mathscr{T}$ des tests T tel que $\{ok(I_1), ok(I_7)\} \in Modes(\mathscr{T})$. Comme I_1 est le super-item de I_7, le remplacement des modes parent nous donne :

$$\begin{aligned} \neg ok(I_1) &\leftrightarrow \neg ok(I_2) \vee \neg ok(I_3) \vee \neg ok(VI_1) \\ \neg ok(I_3) &\leftrightarrow \neg ok(I_6) \vee \neg ok(I_7) \vee \neg ok(VI_2) \end{aligned} \tag{3.3.2}$$

avec $VI_1 = I_1 \setminus \{I_2,\ I_3\}$ et $VI_2 = I_3 \setminus \{I_6,\ I_7\}$

La proposition (3.3.2) est réécrite :

$$\neg ok(I_1) \leftrightarrow \neg ok(I_2) \vee \neg ok(I_6) \vee \neg ok(I_7) \vee \neg ok(VI_2) \vee \neg ok(VI_1) \tag{3.3.3}$$

Dans cet exemple, nous voyons que seulement les parties du système concernant les modes testés sont affinées dans l'analyse diagnostique. Le mode parent $ok(I_2)$ n'est pas testé. Donc, il n'est pas nécessaire de le remplacer dans l'analyse diagnostique.

Durant l'analyse diagnostique avec décompositions itératives, la décomposition d'un item en un ensemble de sous-items peut conduire à un sous-item qui est discriminable avec certains autres items et non-discriminable avec certains autres. L'idée est que, dans l'analyse diagnostique, il ne faut pas analyser en détail les sous-items qui ne sont pas discriminables. Dans le processus de diagnostic, la table de signature sera mise à jour à chaque fois que des nouveaux

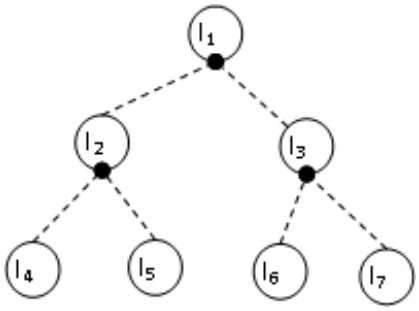


FIG. 3.4 – Calcul des diagnostics

tests sont ajoutés. Supposons que $\mathscr{T}_n$: $Modes(T_n) \rightarrow T_n$ soit la table de signature obtenue à l'itération n dans un processus de diagnostic. Pour assurer la performance des diagnostics, $\mathscr{T}_n$ doit satisfaire deux conditions :

- Il n'existe pas de relations hiérarchiques entre les modes dans $Modes(\mathscr{T}_n)$. Cette condition assure qu'il n'existe pas la relation hiérarchique entre deux modes impliqués dans un diagnostic. Ceci a été discuté précédemment dans la partie de la gestion des relations hiérarchiques des items.
- Il ne faut pas analyser en détail les sous-items qui ne sont pas discriminables.

Calcul effectif des diagnostics (bloc (4c) sur la figure 3.2)

En considérant les défauts multiples dans un système, un diagnostic est donc une conjonction de modes de défaut telle qu'elle peut satisfaire toutes les explications des symptômes. Dans chaque diagnostic, le mode de défaillance de chaque composant défectueux est pris en compte. Autrement dit, étant donné un ensemble de tests inconsistants, la recherche d'un diagnostic consiste à rechercher une conjonction de modes qui vérifie l'expression suivante :

$$\bigwedge_{T_i \in \mathscr{T}} Expl(\top(T_i)) \tag{3.3.4}$$

$\mathscr{T}$ est l'ensemble de tests inconsistants.

Ces diagnostics sont recherchés en utilisant l'algorithme de l'arbre HS de Reiter présenté dans l'annexe A.

remarque 1. $Expl(\top(T_i))$ dans la proposition 3.3.4 est une disjonction logique de modes et est considérée comme *un conflit* dans la construction de l'arbre HS. En réalité, il peut exister à la fois les OUs et les ETs logiques dans $Expl(\top(T_i))$. Dans ce cas, pour appliquer HS-Tree au calcul de diagnostic, la proposition 3.3.4 peut être transformée en forme normale conjonctive représentée par

$$(mode(I_1.1) \vee \cdots \vee mode(I_1.k)) \wedge \cdots \wedge (mode(I_m.1) \vee \cdots \vee mode(I_m.n)) \quad (3.3.5)$$

Chaque clause $(mode(I_m.1) \vee \cdots \vee mode(I_m.n))$ est une disjonction de modes et est considérée comme *un conflit* dans la construction de l'arbre HS.

Théoriquement, nous pouvons transformer la proposition 3.3.4 en forme de la propostion 3.3.5 s'il existe à la fois des OUs et des ETs dans $Expl(\top(T_i))$. Pourtant, cette transformation peut causer des difficultés concernant l'explosion de combinaison dans les calculs. Comme ceci n'est pas le but du travail présenté dans ce manuscrit, nous supposons dans ce travail que $Expl(\top(T_i))$ soit toujours une disjonction de modes. Le traitement du cas où il existe à la fois des OUs et des ETs dans $Expl(\top(T_i))$ sera une perspective à développer.

Les tests consistants sont intégrés comme nous le présenterons dans la suite.

Prise en compte des tests consistants

L'analyse des tests consistants permet d'identifier la liste des fonctions et des ressources en bon fonctionnement, et donc la liste des modes qui ne peuvent pas être présents dans un diagnostic. Cet ensemble est appelé ensemble des modes impossibles et nous notons E_{imp} l'ensemble de ces modes.

L'idée est que, dans la tâche du calcul des diagnostics par l'algorithme de l'arbre HS (annexe A),si un mode impossible est détecté (par la comparaison entre le mode considéré et ensemble des modes impossibles), la construction du diagnostic en cours est arrêtée car elle conduira à un diagnostic impossible. *L'ensemble de modes impossibles* E_{imp} est ainsi géré dans la construction de l'arbre HS de la façon suivante : Si un conflit S est étiqueté à un noeud n sur l'arbre HS (voir en détail la méthode de construction de l'arbre HS à partir de l'ensemble de conflits dans l'annexe A), les branches descendantes seront étiquetées par les éléments impliqués dans S. Supposons que $mode_j(I)$ est étiqueté à une branche. Si $mode_j(I) \in E_{imp}$, alors la branche est fermée car $mode_j(I)$ ne sera pas un diagnostic et il est exonéré.

L'intégration de la gestion des modes non possibles permet de réduire la taille de l'arbre HS.

Classement des diagnostics (bloc (4d) sur la figure 3.2)

Les diagnostics trouvés par l'algorithme pourront donc :
– soit être classés en fonction de la mesure de coïncidence, qui présente l'avantage d'être simple à évaluer, mais qui repose sur la structure des tests et non sur la fiabilité des éléments du système,

– soit être classés en fonction de leur probabilité, qui prend en compte la fiabilité du système, mais nécessite plus d'information de la part de l'expert.

Ceci peut être consulté en détail dans l'annexe A.

3.3.3 Conclusion

Nous avons proposé dans cette section 3.3 l'ensemble des tâches de l'automate pour résoudre le problème formulé par l'expert à chaque itération dans le processus de diagnostic via le modèle FIS (résumé par la section 3.2). Ces tâches sont résumées sur la figure 3.2, page 70. Elles correspondent aux blocs (3) et (4) sur la figure 3.1, page 67.

3.4 Application

Nous allons présenter dans cette partie un exemple du problème de diagnostic interactif en s'appuysant sur FIS. Pour simplifier, nous supposons dans cet exemple que :

– Chaque explication de test a toujours la forme d'une disjonction de mode (selon la remarque 1, page 75)
– Toutes les relations *fonction / ressources de sortie* sont de type *produire* (noté par *p* sur le graphe F/R). Elles peuvent donc être transformées automatiquement sans avoir besoin d'information complémentaire de l'expert (figure 2.20, page 56).

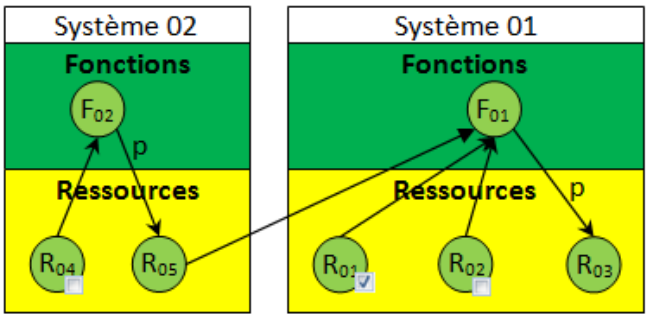
3.4.1 Exemple

Afin d'illustrer comment l'approche proposée correspond à un diagnostic itératif avec les étapes de décomposition consécutives, considérons le problème de diagnostic d'une agence de maintenance qui dépanne *le système d'éclairage automatique* dans un bâtiment. Tout d'abord, un client informe que le système d'allumage automatique dans son bâtiment ne marche pas. À cette étape, un symptôme qui en est déduit : *cfm (Éclairer le bâtiment en fonction du passage des personnes)*. Ce mode de défaillance est très général et est aussi un diagnostic trivial. L'expert commence le processus de diagnostic pour localiser le défaut.

1. **Interaction 1 : Formulation du problème énoncé 1**

   Au fur et à mesure, l'expert décortique le système et le décrit par un outil d'aide au diagnostic qui est conçu en s'appuyant sur FIS. Supposons qu'à l'interaction (1), le système soit décrit partiellement sur la figure 3.5.

   A chaque fois qu'un item est ajouté, deux modes {*ok* ,*cfm* } (ou les états */consistant, inconsistant/*) sont ajoutés automatiquement. Un symptôme est déclaré par l'expert via le logiciel aide au diagnostic.



Système 01

F_{01} : Eclairer le bâtiment en fonction du passage des personnes

R_{01} : Position de la personne

R_{02} : Système électrique-électronique

R_{03} : Lumière autour de la personne

Système 02

F_{02} : Alimenter en énergie électrique

R_{04}: Ressource d'énergie électrique

R_{05} : Courant électrique

FIG. 3.5 – Modèle FIS à l'interaction 1

$$Test \;\; T_1 : \;\; cfm(F_{01}) \tag{3.4.1}$$

2. **Interaction 1 : Transformation du problème énoncé 1**

   Jusqu'ici, le problème énoncé (1) est formulé par le modèle structuro-fonctionnel sur la figure 3.5 et la proposition 3.4.1. L'idée principale de cette étape de transformation est de transformer le graphe F/R en un ensemble de propositions sur les modes tel que nous pouvons exploiter dans l'analyse diagnostique. Le graphe F/R (figure 3.5) peut être traduit en :

$$\begin{aligned} \top(R_{03}) &\leftrightarrow cfm(F_{01}) \\ cfm(F_{01}) &\leftrightarrow \top(R_{01}) \vee cfm(R_{02}) \vee \top(R_{05}) \\ \top(R_{05}) &\leftrightarrow cfm(F_{02}) \\ cfm(F_{02}) &\leftrightarrow cfm(R_{04}) \end{aligned} \tag{3.4.2}$$

   Remarque : Dans cet exemple, nous ne considérons que le cas où l'ensemble d'entrants d'une fonction est complète.

3. **Interaction 1 : Résolution du problème transformé 1**

   De (3.4.1), la propagation en remplaçant des modes dans le sens inverse des flèche nous donne :

$$cfm\ (F_{01}) \rightarrow \top(R_{01}) \vee cfm\ (R_{02}) \vee cfm\ (R_{04}) \tag{3.4.3}$$

Les explications finales des symptômes :

$$Expl(\top(T_1)): \quad \top(R_{01}) \vee cfm\ (R_{02}) \vee cfm\ (R_{04}) \tag{3.4.4}$$

Une table de signature est construite.

	$\perp(R_{01})$	$ok(R_{02})$	$ok(R_{04})$
T_1	1	1	1

TAB. 3.1 – Table de signature donnée à interaction 1

Les diagnostics sont donnés par l'automate

$$\{\top(R_{01})\}, \{cfm\ (R_{02})\}, \{cfm\ (R_{04})\} \tag{3.4.5}$$

Les diagnostics sont classés en un ordre prioritaire grâce à la table de signature (3.1). La signature théorique des défauts est :

Signature théorique	Signature effective	Mesure de coïncidence
$\sigma_T(\top(R_{01})) = (1)$	$\sigma_T^* = (1)$	$\mu_T^c(\top(R_{01})) = 0.00$
$\sigma_T(cfm\ (R_{02})) = (1)$		$\mu_T^c(cfm\ (R_{02})) = 0.00$
$\sigma_T(cfm\ (R_{04})) = (1)$		$\mu_T^c(cfm\ (R_{04})) = 0.00$

TAB. 3.2 – Mesures de coïncidence des défauts à l'interaction 1

L'expert continue à décomposer le système pour tester.

4. **Interaction 2 : Formulation du problème énoncé 2**

   Supposons que *le système électrique-électronique* soit décomposé en deux sous-systèmes : *le circuit électrique* et *le circuit électronique*. Ces deux sous-systèmes sont alimentés en *courant continu* et en *courant alternatif*. Le modèle FIS est enrichi par l'expert sur la figure 3.6. Au-dessous de la figure, les éléments qui sont ajoutés à cette interaction sont écrits en italique.

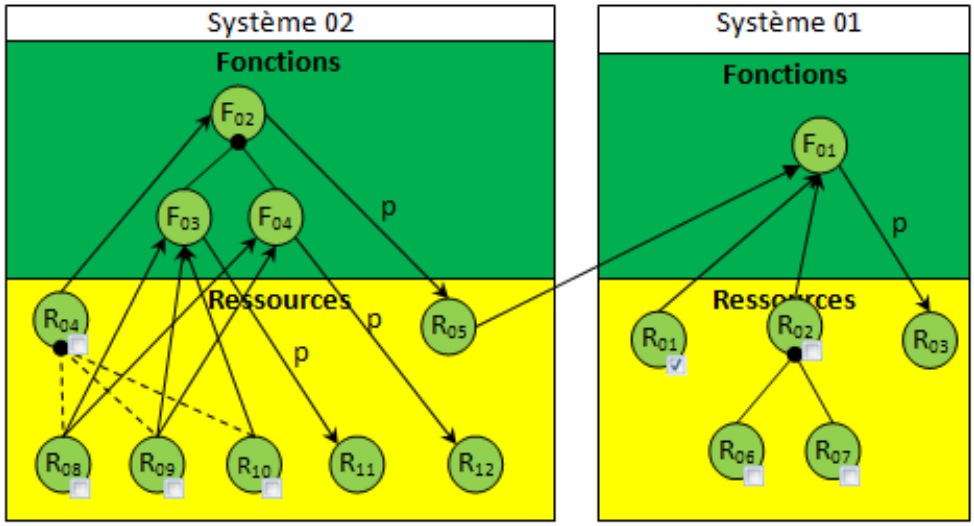
   L'expert mesure le courant alternatif à la sortie du disjoncteur. Une observation est déclarée par l'expert via le logiciel aide au diagnostic.

$$Test\ \ T_2: \perp(R_{12}) \tag{3.4.6}$$

5. **Interaction 2 : Transformation du problème énoncé 2**

   Problème énoncé 2 comporte le modèle structuro-fonctionnel construit par l'expert sur la figure 3.6 et les observations données par les propositions 3.4.1 et 3.4.6.

   L'étape de transformation complète automatiquement des abstractions partielles en ajoutant des items virtuels : $IV_{2.1} = R_{04} \setminus \{R_{08}, R_{09}, R_{10}\}$.



Système 01

F01 : Eclairer le bâtiment en fonction du passage des personnes

R01 : Position de la personne

R02 : Système électrique-électronique

R03 : Lumière autour de la personne

R06 : *Circuit électronique*

R07 : *Circuit électrique*

Système 02

F02 : Alimenter en énergie électrique

R04: Ressource d'énergie électrique

R05 : Courant électrique

F03 : *Alimenter en DC*

F04 : *Alimenter en AC*

R08: *Secteur électrique*

R09: *Disjoncteurs*

R10 : *Convertisseur AC/DC*

R11 : *Courant DC*

R12 : *Courant AC*

FIG. 3.6 – Modèle FIS à l'interaction 2

Transformation du modèle structuro-fonctionnel donné sur la figure 3.6 en un ensemble de propositions logiques. Cet ensemble comporte les propositions logiques 3.4.2 et 3.4.7.

$$\begin{aligned} cfm(R_{02}) &\leftrightarrow cfm(R_{06}) \vee cfm(R_{07}) \\ cfm(F_{02}) &\leftrightarrow cfm(F_{03}) \vee cfm(F_{04}) \\ cfm(R_{04}) &\leftrightarrow cfm(R_{08}) \vee cfm(R_{09}) \vee cfm(R_{10}) \vee cfm(IV_{2.1}) \\ \top(R_{12}) &\leftrightarrow cfm(F_{04}) \\ cfm(F_{04}) &\leftrightarrow cfm(R_{08}) \vee cfm(R_{09}) \\ cfm(R_{10}) &\leftrightarrow cfm(F_{03}) \\ cfm(F_{03}) &\leftrightarrow cfm(R_{08}) \vee cfm(R_{09}) \vee cfm(R_{10}) \end{aligned} \tag{3.4.7}$$

6. **Interaction 2 : Résolution du problème transformé 2**

De la proposition 3.4.6, la propagation en remplaçant des modes dans le sens inverse des flèches nous donne :

$$\perp(R_{12}) \rightarrow ok\ (R_{08}) \wedge ok\ (R_{09}) \tag{3.4.8}$$

En comparaison entre les propositions 3.4.3 et 3.4.8, nous voyons qu'il existe des relations hiérarchiques entre R_{04} et R_{08}. Le mode $cfm\ (R_{04})$ impliqué dans 3.4.3 doit être remplacé par les modes de ses sous-ressources jusqu'à ce qu'il n'existe plus de relations hiérarchiques dans l'ensemble des explications des symptômes. Le remplacement nous donne des explications finales :

$$\begin{aligned} Expl(\top(T_1)):\ \ & \top(R_{01}) \vee cfm\ (R_{02}) \vee cfm(R_{08}) \ldots \\ & \cdots \vee cfm(R_{09}) \vee cfm(R_{10}) \vee cfm(IV_{2.1}) \\ & Expl(\perp(T_2)):\ \ ok\ (R_{08}) \wedge ok\ (R_{09}) \end{aligned} \tag{3.4.9}$$

La table de signature est construite.

	$\perp(R_{01})$	$ok\ (R_{02})$	$ok\ (R_{08})$	$ok\ (R_{09})$	$ok\ (R_{10})$	$ok\ (IV_{2.1})$
T_1	1	1	1	1	1	1
T_2	0	0	1	1	0	0

TAB. 3.3 – Table de signature donnée à interaction 2

Les diagnostics sont donnés par l'automate

$$\{\top(R_{01})\}, \{cfm\ (R_{02})\}, \{cfm\ (R_{10})\}, \{cfm\ (IV_{2.1})\} \tag{3.4.10}$$

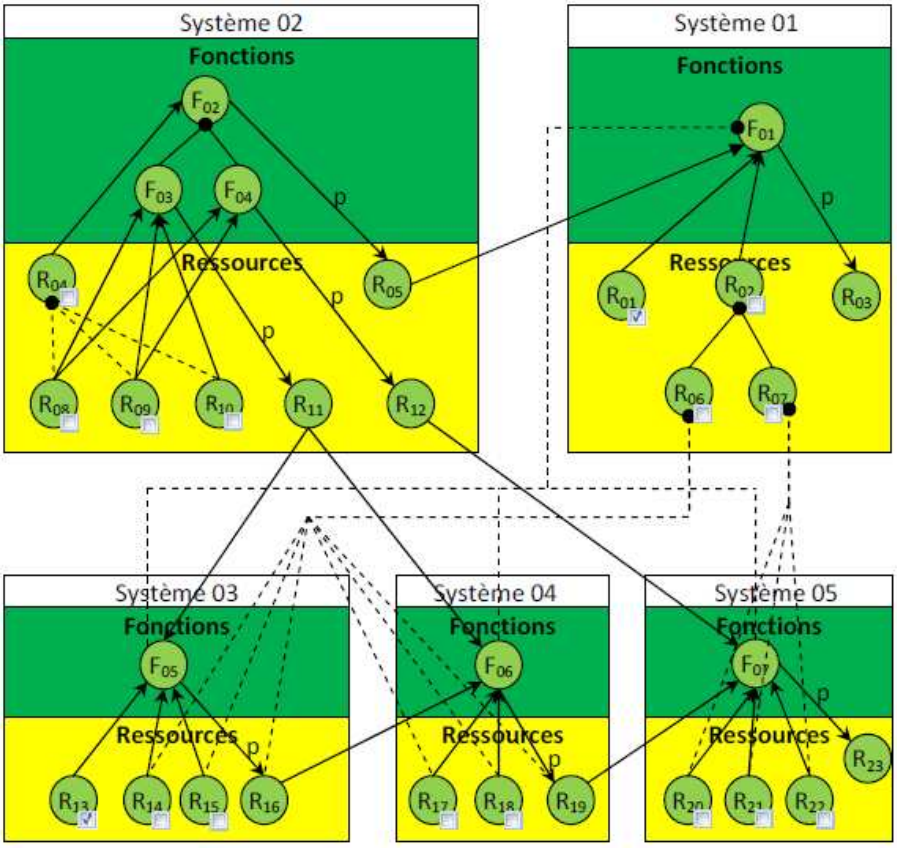
Les diagnostics sont classés en un ordre prioritaire grâce à la table de signature (3.3). La signature théorique des défauts est :

Signature théorique	Signature effective	Mesure de coïncidence
$\sigma_T(\top(R_{01}) = (1\ \ 0)$	$\sigma_T^* = (1\ \ 0)$	$\mu_T^c(\top(R_{01})) = 0.00$
$\sigma_T(cfm\ (R_{02}) = (1\ \ 0)$		$\mu_T^c(cfm\ (R_{02})) = 0.00$
$\sigma_T(cfm\ (R_{10}) = (1\ \ 0)$		$\mu_T^c(cfm\ (E_{10})) = 0.00$
$\sigma_T(cfm\ (IV_{2.1}) = (1\ \ 0)$		$\mu_T^c(cfm\ (IV_{2.1})) = 0.00$

TAB. 3.4 – Mesure de coïncidence des défauts à interaction 2

7. **Interaction 3 : Formulation du problème énoncé 3**

   L'expert passe au couloir pour faire un test, la lampe au couloir ne s'allume pas. Il continue à décortiquer et à décrire sa connaissance. Le modèle FIS est enrichi par l'expert sur la figure 3.7



Système 01

F_{01} : Eclairer le bâtiment en fonction du passage des personnes
R_{01} : Position de la personne
R_{02} : Système électrique-électronique
R_{03} : Lumière autour de la personne
R_{06} : Circuit électronique
R_{07} : Circuit électrique

Système 02

F_{02} : Alimenter en énergie électrique
R_{04}: Ressource d'énergie électrique
R_{05} : Courant électrique
F_{03} : Alimenter en DC
F_{04} : Alimenter en AC
R_{08}: Secteur électrique
R_{09}: Disjoncteurs
R_{10} : Convertisseur AC/DC
R_{11} : Courant DC
R_{12} : Courant AC

Système 03

F_{05} : Détecter le passage des personnes dans le couloir
R_{13}: Passage de la personne dans le couloir
R_{14}: Capteurs au couloir
R_{15}: Connections physiques capteurs au couloir/centre de traitement
R_{16} : Signal [personne présente dans le couloir]

Système 04

F_{06} : Traiter les informations du couloir
R_{17} : Microcontrôleur
R_{18} : Connections physiques centre de traitement/ relais du lampe au couloir
R_{19} : Signal de command lampe au couloir

Système 05

F_{07}: Allumer lampe au couloir
R_{20} : Câbles électriques
R_{21} : Relais de la lampe au couloir
R_{22} : Lampe au couloir
R_{23} : Lumière au couloir

FIG. 3.7 – Modèle FIS à l'interaction 3

Une observation est déclarée par l'expert via le logiciel aide au diagnostic.

$$Test \;\; T_3 : \;\; cfm\,(F_{07}) \tag{3.4.11}$$

8. **Interaction 3 : Transformation du problème énoncé 3**
   Problème énoncé 3 comporte le modèle structuro-fonctionnel construit par l'expert sur la figure 3.7 et les observations exprimées par les propositions : 3.4.1, 3.4.6 et 3.4.11.
   L'étape de transformation complète automatiquement des abstractions partielles en ajoutant des items virtuels

$$IV_{3.1} = R_{04} \setminus \{R_{08}, R_{09}, R_{10}\} \tag{3.4.12}$$
$$IV_{3.2} = R_{06} \setminus \{R_{17}, R_{18}, R_{19}, R_{14}, R_{15}, R_{16}\} \tag{3.4.13}$$
$$IV_{3.4} = R_{07} \setminus \{R_{20}, R_{21}, R_{22}\} \tag{3.4.14}$$
$$\tag{3.4.15}$$

   Transformation du modèle structuro-fonctionnel donné sur la figure 3.7 en un ensemble de propositions logiques donné par 3.4.16 :

$$\begin{aligned}
cfm(R_{03}) &\leftrightarrow cfm(F_{01}) \\
cfm(F_{01}) &\leftrightarrow \top(R_{01}) \vee cfm(R_{02}) \vee \top(R_{05}) \\
\top(R_{05}) &\leftrightarrow cfm(F_{02}) \\
cfm(F_{02}) &\leftrightarrow cfm(R_{04}) \\
\top(R_{12}) &\leftrightarrow cfm(F_{04}) \\
cfm(F_{04}) &\leftrightarrow cfm(R_{08}) \vee cfm(R_{09}) \\
cfm(F_{03}) &\leftrightarrow cfm(R_{08}) \vee cfm(R_{09}) \vee cfm(R_{10}) \\
cfm(R_{11}) &\leftrightarrow cfm(F_{03}) \\
\top(R_{23}) &\leftrightarrow cfm\,(F_{07}) \\
cfm\,(F_{07}) &\leftrightarrow \top(R_{12}) \vee \top(R_{19}) \vee cfm\,(R_{20}) \ldots \\
\cdots &\vee cfm\,(R_{21}) \vee cfm\,(R_{22}) \\
\top(R_{19}) &\leftrightarrow cfm\,(F_{06}) \\
cfm\,(F_{06}) &\leftrightarrow \top(R_{11}) \vee \top(R_{16}) \vee cfm\,(R_{17}) \vee cfm\,(R_{18}) \\
\top(R_{16}) &\leftrightarrow cfm\,(F_{05}) \\
cfm\,(F_{05}) &\leftrightarrow \top(R_{11}) \vee \top(R_{13}) \vee cfm\,(R_{14}) \vee cfm\,(R_{15}) \\
cfm(R_{02}) &\leftrightarrow cfm(R_{06}) \vee cfm(R_{07}) \\
cfm(F_{02}) &\leftrightarrow cfm(F_{03}) \vee cfm(F_{04}) \\
cfm(R_{04}) &\leftrightarrow cfm(R_{08}) \vee cfm(R_{09}) \vee cfm(R_{10}) \vee cfm(IV_{3.1}) \\
cfm\,(R_{06}) &\leftrightarrow cfm\,(R_{14}) \vee cfm\,(R_{15}) \vee \top(R_{16}) \vee cfm\,(R_{17}) \ldots \\
\cdots &\vee cfm\,(R_{18}) \vee \top(R_{19}) \vee cfm\,(IV_{3.2}) \\
cfm\,(R_{07}) &\leftrightarrow cfm\,(R_{20}) \vee cfm\,(R_{21}) \vee cfm\,(R_{22}) \vee cfm\,(IV_{3.3})
\end{aligned} \tag{3.4.16}$$

9. **Interaction 3 : Résolution du problème transformé 3**

De 3.4.11, la propagation en remplaçant des modes dans le sens inverse des flèches nous donne :

$$\begin{aligned} cfm\ (F_{07}) \rightarrow cfm\ (R_{08}) \vee cfm\ (R_{09}) \vee cfm\ (R_{20}) \vee cfm\ (R_{21}) \vee cfm\ (R_{22}) \ldots \\ \cdots \vee cfm\ (R_{10}) \vee cfm\ (R_{17}) \vee cfm\ (R_{18}) \vee \top(R_{13}) \ldots \\ \cdots \vee cfm\ (R_{14}) \vee cfm\ (R_{15}) \end{aligned} \tag{3.4.17}$$

En remplaçant les modes des items-parents dans les propositions 3.4.3, 3.4.8, 3.4.17, nous avons des explications finales :

$$\begin{aligned} Expl(\top(T_1)) :\ & cfm\ (R_{20}) \vee cfm\ (R_{21}) \vee cfm(R_{22}) \vee cfm(R_{17}) \ldots \\ & \cdots \vee cfm(R_{18}) \vee \top(R_{13}) \vee cfm(R_{14}) \vee cfm(R_{15}) \vee cfm(R_{08}) \ldots \\ & \cdots \vee cfm(R_{09}) \vee cfm(R_{10}) \vee cfm(IV_{3.1}) \vee cfm(IV_{3.2}) \vee cfm(IV_{3.3}) \\ Expl(\bot(T_2)) :\ & ok\ (R_{08}) \wedge ok\ (R_{09}) \\ Expl(\top(T_3)) :\ & cfm\ (R_{20}) \vee cfm\ (R_{21}) \vee cfm(R_{22}) \vee cfm(R_{17}) \ldots \\ & \cdots \vee cfm(R_{18}) \vee \top(R_{13}) \vee cfm(R_{14}) \vee cfm(R_{15}) \vee cfm(R_{08}) \ldots \\ & \cdots \vee cfm(R_{09}) \vee cfm(R_{10}) \end{aligned} \tag{3.4.18}$$

La table de signature (3.5) est construite

	$ok(R_{08})$	$ok(R_{09})$	$ok(R_{10})$	$\bot(R_{13})$	$ok(R_{14})$	$ok(R_{15})$	$ok(R_{17})$	$ok(R_{18})$
T_1	1	1	1	1	1	1	1	1
T_2	1	1	0	0	0	0	0	0
T_3	1	1	1	1	1	1	1	1

$ok(R_{20})$	$ok(R_{21})$	$ok(R_{22})$	$ok(IV_{3.1})$	$ok(IV_{3.2})$	$ok(IV_{3.3})$
1	1	1	1	1	1
0	0	0	0	0	0
1	1	1	0	0	0

TAB. 3.5 – Table de signature donnée à interaction 3

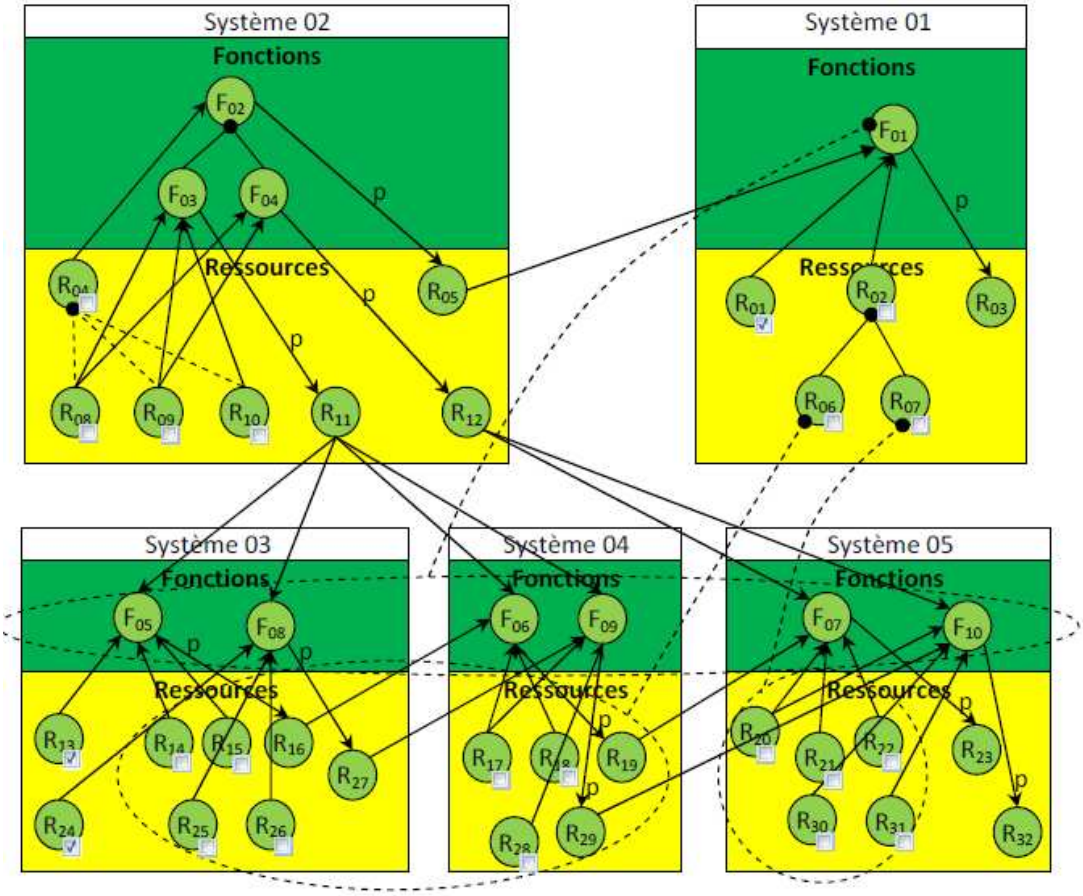
Les diagnostics sont donnés par l'automate

$\{cfm\ (R_{10})\}$, $\{\top(R_{13})\}$, $\{cfm\ (R_{14})\}$, $\{cfm\ (R_{15})\}$, $\{cfm\ (R_{17})\}$, $\{cfm\ (R_{18})\}$, $\{cfm\ (R_{20})\}$, $\{cfm\ (R_{21})\}$, $\{cfm\ (R_{22})\}$

Les diagnostics sont classés en un ordre prioritaire grâce à la table de signature 3.5. La signature théorique des défauts est :

10. **Interaction 4 : Formulation du problème énoncé 4**

L'expert passe au salon pour faire un test, la lampe au salon ne s'allume pas. Il continue à décortiquer et à décrire sa connaissance. Le modèle FIS est enrichi par l'expert sur la figure 3.8.



Remarques : Pour être plus visible, l'ensemble des sous-items d'un super item est entouré par un cercle pointillé.

Système 01

F_{01} : Eclairer le bâtiment en fonction du passage des personnes
R_{01} : Position de la personne
R_{02} : Système électrique-électronique
R_{03} : Lumière autour de la personne
R_{06} : Circuit électronique
R_{07} : Circuit électrique

Système 02

F_{02} : Alimenter en énergie électrique
R_{04}: Ressource d'énergie électrique
R_{05} : Courant électrique
F_{03} : Alimenter en DC
F_{04} : Alimenter en AC
R_{08}: Secteur électrique
R_{09}: Disjoncteurs
R_{10} : Convertisseur AC/DC
R_{11} : Courant DC
R_{12} : Courant AC

Système 03

F_{05} : Détecter le passage des personnes dans le couloir
R_{13}: Passage de la personne dans le couloir
R_{14}: Capteurs au couloir
R_{15}: Connections physiques capteurs au couloir/centre de traitement
R_{16} : Signal [personne présente dans le couloir]
F_{08} : Détecter le passage des personnes dans le salon
R_{24}: Passage de la personne dans le salon
R_{25}: Capteurs au salon
R_{26}: Connections physiques capteurs au salon/centre de traitement
R_{27} : Signal [personne présente dans le salon]

Système 04

F_{06} : Traiter les informations du couloir
R_{17} : Microcontrôleur
R_{18} : Connections physiques centre de traitement/relais du lampe au couloir
R_{19} : Signal de command pour lampe au couloir
F_{09} : Traiter les informations du salon
R_{28} : Connections physiques centre de traitement/relais du lampe au salon
R_{29} : Signal de command pour lampe au salon

Système 05

F_{07}: Allumer lampe au couloir
R_{20} : Câbles électriques
R_{21} : Relais de la lampe au couloir
R_{22} : Lampe au couloir
R_{23} : Lumière au couloir
F_{10}: Allumer lampe au salon
R_{24} : Relais de la lampe au salon
R_{25} : Lampe au salon
R_{32} : Lumière au salon

FIG. 3.8 – Modèle FIS à l'interaction 4

Signature théorique	Signature effective	Mesure de coïncidence
$\sigma_T(cfm\ (R_{10})) = (1\ \ 0\ \ 1)$	$\sigma_T^* = (1\ \ 0\ \ 1)$	$\mu_T^c(cfm\ (R_{10})) = 0.00$
$\sigma_T(\top(R_{13})) = (1\ \ 0\ \ 1)$		$\mu_T^c(\top(R_{13})) = 0.00$
$\sigma_T(cfm\ (R_{14})) = (1\ \ 0\ \ 1)$		$\mu_T^c(cfm\ (R_{14})) = 0.00$
$\sigma_T(cfm\ (R_{15})) = (1\ \ 0\ \ 1)$		$\mu_T^c(cfm\ (R_{15})) = 0.00$
$\sigma_T(cfm\ (R_{17})) = (1\ \ 0\ \ 1)$		$\mu_T^c(cfm\ (R_{17})) = 0.00$
$\sigma_T(cfm\ (R_{18})) = (1\ \ 0\ \ 1)$		$\mu_T^c(cfm\ (R_{18})) = 0.00$
$\sigma_T(cfm\ (R_{20})) = (1\ \ 0\ \ 1)$		$\mu_T^c(cfm\ (R_{20})) = 0.00$
$\sigma_T(cfm\ (R_{21})) = (1\ \ 0\ \ 1)$		$\mu_T^c(cfm\ (R_{21})) = 0.00$
$\sigma_T(cfm\ (R_{22})) = (1\ \ 0\ \ 1)$		$\mu_T^c(cfm\ (R_{22})) = 0.00$

TAB. 3.6 – Mesure de coïncidence des défauts à interaction 3

Une observation est déclarée par l'expert via le logiciel aide au diagnostic.

$$Test\ \ T_4 :\ \ cfm\ (F_{10}) \tag{3.4.19}$$

11. **Interaction 4 : Transformation du problème énoncé 4** Problème annoncé 4 comporte le modèle structuro-fonctionnel construit par l'expert sur la figure 3.8 et les observations exprimées par les propositions : 3.4.1, 3.4.6, 3.4.11 et 3.4.19.

    L'étape de transformation complète automatiquement des abstractions partielles en ajoutant des items virtuels

$$IV_{4.1} = R_{04} \setminus \{R_{08}, R_{09}, R_{10}\}$$
$$IV_{4.2} = R_{06} \setminus \{R_{17}, R_{18}, R_{19}, R_{14}, R_{15}, R_{16}, R_{25}, R_{26}, R_{27}, R_{28}, R_{29}\}$$
$$IV_{4.3} = R_{07} \setminus \{R_{20}, R_{21}, R_{22}, R_{30}, R_{31}\}$$

    Transformation du modèle structuro-fonctionnel donné sur la figure 3.8 en un ensemble de propositions logiques donné par 3.4.20 :

$$
\begin{aligned}
cfm(R_{03}) &\leftrightarrow cfm(F_{01})\\
cfm(F_{01}) &\leftrightarrow \top(R_{01}) \vee cfm(R_{02}) \vee \top(R_{05})\\
\top(R_{05}) &\leftrightarrow cfm(F_{02})\\
cfm(F_{02}) &\leftrightarrow cfm(R_{04})\\
\top(R_{12}) &\leftrightarrow cfm(F_{04})\\
cfm(F_{04}) &\leftrightarrow cfm(R_{08}) \vee cfm(R_{09})\\
cfm(F_{03}) &\leftrightarrow cfm(R_{08}) \vee cfm(R_{09}) \vee cfm(R_{10})\\
cfm(R_{11}) &\leftrightarrow cfm(F_{03})\\
\top(R_{23}) &\leftrightarrow cfm\ (F_{07})\\
cfm\ (F_{07}) &\leftrightarrow \top(R_{12}) \vee \top(R_{19}) \vee cfm\ (R_{20}) \ldots\\
&\cdots \vee cfm\ (R_{21}) \vee cfm\ (R_{22})\\
\top(R_{19}) &\leftrightarrow cfm\ (F_{06})\\
cfm\ (F_{06}) &\leftrightarrow \top(R_{11}) \vee \top(R_{16}) \vee cfm\ (R_{17}) \vee cfm\ (R_{18})\\
\top(R_{16}) &\leftrightarrow cfm\ (F_{05})\\
cfm\ (F_{05}) &\leftrightarrow \top(R_{11}) \vee \top(R_{13}) \vee cfm\ (R_{14}) \vee cfm\ (R_{15})\\
\top(R_{32}) &\leftrightarrow cfm\ (F_{10})\\
cfm\ (F_{10}) &\leftrightarrow \top(R_{12}) \vee \top(R_{29}) \vee cfm\ (R_{20}) \vee cfm\ (R_{30}) \vee cfm\ (R_{31})\\
\top(R_{29}) &\leftrightarrow cfm\ (F_{09})\\
cfm\ (F_{09}) &\leftrightarrow \top(R_{11}) \vee \top(R_{27}) \vee cfm\ (R_{17}) \vee cfm\ (R_{28})\\
\top(R_{27}) &\leftrightarrow cfm\ (F_{08})\\
cfm\ (F_{08}) &\leftrightarrow \top(R_{11}) \vee \top(R_{24}) \vee cfm\ (R_{25}) \vee cfm\ (R_{26})\\
cfm(R_{02}) &\leftrightarrow cfm(R_{06}) \vee cfm(R_{07})\\
cfm(F_{02}) &\leftrightarrow cfm(F_{03}) \vee cfm(F_{04})\\
cfm(R_{04}) &\leftrightarrow cfm(R_{08}) \vee cfm(R_{09}) \vee cfm(R_{10}) \vee cfm(IV_{3.1})\\
cfm\ (R_{06}) &\leftrightarrow cfm\ (R_{14}) \vee cfm\ (R_{15}) \vee \top(R_{16}) \vee cfm\ (R_{17}) \ldots\\
&\cdots \vee cfm\ (R_{18}) \vee \top(R_{19}) \vee cfm\ (R_{25}) \vee cfm\ (R_{26}) \vee \top(R_{27}) \ldots\\
&\cdots \vee cfm\ (R_{28}) \vee \top(R_{29}) \vee cfm\ (IV_{4.2})\\
cfm\ (R_{07}) &\leftrightarrow cfm\ (R_{20}) \vee cfm\ (R_{21}) \vee cfm\ (R_{22}) \vee cfm\ (R_{30}) \ldots\\
&\cdots \vee cfm\ (R_{31}) \vee cfm\ (IV_{4.3})
\end{aligned}
\tag{3.4.20}
$$

12. **Interaction 4 : Résolution du problème transformé 4**

De 3.4.19, la propagation en remplaçant des modes dans le sens inverse des flèches nous

donne :

$$\begin{aligned} cfm\ (F_{10}) \rightarrow cfm\ (R_{08}) \vee cfm\ (R_{09}) \vee cfm\ (R_{20}) \vee cfm\ (R_{30}) \ldots \\ \cdots \vee cfm\ (R_{31}) \vee cfm\ (R_{10}) \vee cfm\ (R_{17}) \vee cfm\ (R_{28}) \vee \top(R_{24}) \ldots \\ \cdots \vee cfm\ (R_{25}) \vee cfm\ (R_{26}) \end{aligned} \tag{3.4.21}$$

En remplaçant les modes des items-parents dans 3.4.3, 3.4.8, 3.4.17 et 3.4.21, nous avons des explications finales :

$$\begin{aligned} Expl(\top(T_1)) :\ & cfm\ (R_{20}) \vee cfm\ (R_{21}) \vee cfm(R_{22}) \vee cfm\ (R_{30}) \ldots \\ & \cdots \vee cfm(R_{31}) \vee cfm(R_{17}) \vee cfm(R_{18}) \vee \top(R_{13}) \vee cfm(R_{14}) \ldots \\ & \cdots \vee cfm(R_{15}) \vee cfm(R_{28}) \vee \top(R_{24}) \vee cfm(R_{25}) \vee cfm(R_{26}) \ldots \\ & \cdots \vee cfm(R_{08}) \vee cfm(R_{09}) \vee cfm(R_{10}) \vee cfm(IV_{4.1}) \vee cfm(IV_{4.2}) \ldots \\ & \cdots \vee cfm(IV_{4.3}) \\ Expl(\bot(T_2)) :\ & ok\ (R_{08}) \wedge ok\ (R_{09}) \\ Expl(\top(T_3)) :\ & cfm\ (R_{20}) \vee cfm\ (R_{21}) \vee cfm(R_{22}) \vee cfm(R_{17}) \ldots \\ & \cdots \vee cfm(R_{18}) \vee \top(R_{13}) \vee cfm(R_{14}) \vee cfm(R_{15}) \vee cfm(R_{08}) \ldots \\ & \cdots \vee cfm(R_{09}) \vee cfm(R_{10}) \\ Expl(\top(T_4)) :\ & cfm\ (R_{20}) \vee cfm\ (R_{30}) \vee cfm(R_{31}) \vee cfm(R_{17}) \ldots \\ & \cdots \vee cfm(R_{28}) \vee \top(R_{24}) \vee cfm(R_{25}) \vee cfm(R_{26}) \vee cfm(R_{08}) \ldots \\ & \cdots \vee cfm(R_{09}) \vee cfm(R_{10}) \end{aligned} \tag{3.4.22}$$

La table de signature 3.7 est construite.

Les diagnostics sont donnés par l'automate

$\{cfm\ (R_{10})\}, \{cfm\ (R_{17})\}, \{cfm\ (R_{20})\}$
$\{cfm\ (R_{14}) \wedge cfm\ (R_{30})\}, \quad \{cfm\ (R_{14}) \wedge cfm\ (R_{31})\}, \quad \{cfm\ (R_{14}) \wedge cfm\ (R_{28})\},$
$\{cfm\ (R_{14}) \wedge \top(R_{24})\}, \quad \{cfm\ (R_{14}) \wedge cfm\ (R_{25})\}, \quad \{cfm\ (R_{14}) \wedge cfm\ (R_{26})\}$
$\{cfm\ (R_{15}) \wedge cfm\ (R_{30})\}, \quad \{cfm\ (R_{15}) \wedge cfm\ (R_{31})\}, \quad \{cfm\ (R_{15}) \wedge cfm\ (R_{28})\},$
$\{cfm\ (R_{15}) \wedge \top(R_{24})\}, \quad \{cfm\ (R_{15}) \wedge cfm\ (R_{25})\}, \quad \{cfm\ (R_{15}) \wedge cfm\ (R_{26})\}$
$\{\top(R_{13}) \wedge cfm\ (R_{30})\}, \ \{\top(R_{13}) \wedge cfm\ (R_{31})\}, \ \{\top(R_{13}) \wedge cfm\ (R_{28})\}, \ \{\top(R_{13}) \wedge \top(R_{24})\}, \ \{\top(R_{13}) \wedge cfm\ (R_{25})\}, \ \{\top(R_{13}) \wedge cfm\ (R_{26})\} \ \{cfm\ (R_{21}) \wedge cfm\ (R_{30})\},$
$\{cfm\ (R_{21}) \wedge cfm\ (R_{31})\}, \quad \{cfm\ (R_{21}) \wedge cfm\ (R_{28})\}, \quad \{cfm\ (R_{21}) \wedge \top(R_{24})\},$
$\{cfm\ (R_{21}) \wedge cfm\ (R_{25})\}, \quad \{cfm\ (R_{21}) \wedge cfm\ (R_{26})\} \quad \{cfm\ (R_{22}) \wedge cfm\ (R_{30})\},$
$\{cfm\ (R_{22}) \wedge cfm\ (R_{31})\}, \quad \{cfm\ (R_{22}) \wedge cfm\ (R_{28})\}, \quad \{cfm\ (R_{22}) \wedge \top(R_{24})\},$
$\{cfm\ (R_{22}) \wedge cfm\ (R_{25})\}, \quad \{cfm\ (R_{22}) \wedge cfm\ (R_{26})\} \quad \{cfm\ (R_{18}) \wedge cfm\ (R_{30})\},$
$\{cfm\ (R_{18}) \wedge cfm\ (R_{31})\}, \quad \{cfm\ (R_{18}) \wedge cfm\ (R_{28})\}, \quad \{cfm\ (R_{18}) \wedge \top(R_{24})\},$
$\{cfm\ (R_{18}) \wedge cfm\ (R_{25})\}, \{cfm\ (R_{18}) \wedge cfm\ (R_{26})\}$

Dans l'interaction 4, nous avons des défauts multiples. Cela signifie que deux défauts peuvent se manifester en même temps dans le système. Aucun défaut multiple n'était pas apparu dans les interactions 1,2 et 3. L'algorithme de l'arbre-HS

	$ok(R_{08})$	$ok(R_{09})$	$ok(R_{10})$	$\perp(R_{13})$	$ok(R_{14})$	$ok(R_{15})$	$ok(R_{17})$	$ok(R_{18})$
T_1	1	1	1	1	1	1	1	1
T_2	1	1	0	0	0	0	0	0
T_3	1	1	1	1	1	1	1	1
T_4	1	1	1	0	0	0	1	0

$ok(R_{20})$	$ok(R_{21})$	$ok(R_{22})$	$\perp(R_{24})$	$ok(R_{25})$	$ok(R_{26})$	$ok(R_{28})$
1	1	1	1	1	1	1
0	0	0	0	0	0	0
1	1	1	0	0	0	0
1	0	0	1	1	1	1

$ok(R_{30})$	$ok(R_{31})$	$ok(IV_{4.1})$	$ok(IV_{4.2})$	$ok(IV_{4.3})$	$ok(IV_{4.4})$
1	1	1	1	1	1
0	0	0	0	0	0
0	0	0	0	0	0
1	1	0	0	0	0

TAB. 3.7 – Table de signature donnée à interaction 4

(Annexe A) permet de calculer les défauts multiples. Un défaut multiple est représenté par une conjonction de modes tel que l'ensemble des modes impliqués dans cette conjonction permet d'expliquer l'ensemble des symptômes constatés. Prenons l'exemple d'un diagnostic qui exprime un défaut multiple donné dans cette interaction : $\{cfm\ (R_{14}) \wedge cfm\ (R_{30})\}$. Nous voyons que $cfm\ (R_{14})$ permet d'expliquer des tests inconsistants $\top(T_1)$ et $\top(T_3)$ car $cfm\ (R_{14})$ appartient à l'ensemble des modes impliqués dans $Expl(\top(T_1))$ et $Expl(\top(T_3))$. Néanmoins, $cfm\ (R_{14})$ n'est pas suffisant car il n'explique pas $\top(T_4)$. $cfm\ (R_{30})$ permet lui d'expliquer $\top(T_1)$ et $\top(T_4)$. La conjonction $\{cfm\ (R_{14}) \wedge cfm\ (R_{30})\}$ explique donc l'ensemble des symptômes obtenus à cette interaction. L'utilisation de l'arbre-HS permet de chercher toutes les combinaisons minimales possibles pour expliquer l'ensemble des symptômes en évitant l'explosion combinatoire des calculs.

Les diagnostics sont classés en un ordre prioritaire grâce à la table de la signature 3.7. La signature théorique des défauts est représentée dans la table 3.8.

Et ainsi de suite, l'expert continue la procédure de diagnostic en décomposant au fur et à mesure le système. La procédure s'arrête quand l'expert peut localiser le défaut à un niveau de détail souhaité.

3.4.2 Outil logiciel

XRisk est un outil logiciel développé à partir du modèle FIS. Il a été développé dans un premier temps pour réaliser des analyses de risques. XRisk permet à l'utilisateur de modéliser

Signature théorique	Signature effective	Mesure de coïncidence
$\sigma_T(cfm\ (R_{10})) = (1\ \ 0\ \ 1\ \ 1)$	$\sigma_T^* = (1\ \ 0\ \ 1\ \ 1)$	$\mu_T^c(cfm\ (R_{10})) = 0.00$
$\sigma_T(cfm\ (R_{17})) = (1\ \ 0\ \ 1\ \ 1)$		$\mu_T^c(cfm\ (R_{17})) = 0.00$
$\sigma_T(cfm\ (R_{20})) = (1\ \ 0\ \ 1\ \ 1)$		$\mu_T^c(cfm\ (R_{20})) = 0.00$
$\sigma_T(cfm\ (R_{14}) \wedge cfm\ (R_{30})) = (1\ \ 0\ \ 1\ \ 1)$		$\mu_T^c(cfm\ (R_{14}) \wedge cfm\ (R_{30})) = 0.00$
$\sigma_T(cfm\ (R_{14}) \wedge cfm\ (R_{31})) = (1\ \ 0\ \ 1\ \ 1)$		$\mu_T^c(cfm\ (R_{14}) \wedge cfm\ (R_{31})) = 0.00$
...	...	...
$\sigma_T(cfm\ (R_{18}) \wedge cfm\ (R_{26})) = (1\ \ 0\ \ 1\ \ 1)$		$\mu_T^c(cfm\ (R_{18}) \wedge cfm\ (R_{26})) = 0.00$

TAB. 3.8 – Mesure de coïncidence des défauts à interaction 4

les processus et les systèmes industriels en s'appuyant sur le modèle structuro-fonctionnel FIS. Il implante la démarche développée dans (Flaus et Pham [2010]) et permet d'illustrer les résultats de ce chapitre en demandant à l'expert de réaliser la modélisation dysfonctionnelle suivant les règles présentées dans le chapitre 2.

Ceci donne à l'utilisateur un nouvel outil pour réaliser l'analyse diagnostique. L'analyse diagnostique est une procédure où le modèle du système est construit au fur et à mesure à différents niveaux de détail en s'appuyant sur la modélisation structuro-fonctionnelle FIS.

La figure 3.9 présente l'interface de XRisk qui permet de construire le modèle structuro-fonctionnel du système. (a) est un système. Les fonctions et les ressources des sous-systèmes impliqués dans les sous-systèmes peuvent être vues au niveau du super-système. (b) sont les fonctions impliquées dans le système. (c) sont les ressources impliquées dans le système. (d) représente la hiérarchie des systèmes. En cliquant sur chaque item, une fenêtre apparaît qui permet de donner l'identifiant (ID) et le nom (f) de l'item et de déterminer l'item parent (e).

La figure 3.10 présente l'interface de XRisk qui permet à l'expert de déclarer les tests consistants et les tests inconsistants. (a) est l'identification du test. (b) est la liste de modes impliqués dans l'explication d'un test consistant. Il existe un ET logique entre ces modes. (c) est la liste de modes impliqués dans l'explication dans un test inconsistant. Il existe un OU logique entre ces modes. (d) est l'ensemble des modes qui sont complètement testés. Le bouton *edit* (e) permet d'ajouter et de supprimer les éléments dans (b), (c), (d). Un élément est ajouté s'il est coché.

La figure 3.11 présente l'interface de XRisk qui permet de calculer des diagnostics à partir des tests consistants (c), inconsistants (b), et les modes complètement testés (a). Le bouton *edit* (d) permet d'ajouter et de supprimer les éléments dans (a), (b), (c). (d) est la liste des diagnostics trouvés. Par défaut, la probabilité de chaque mode est égale à 1.

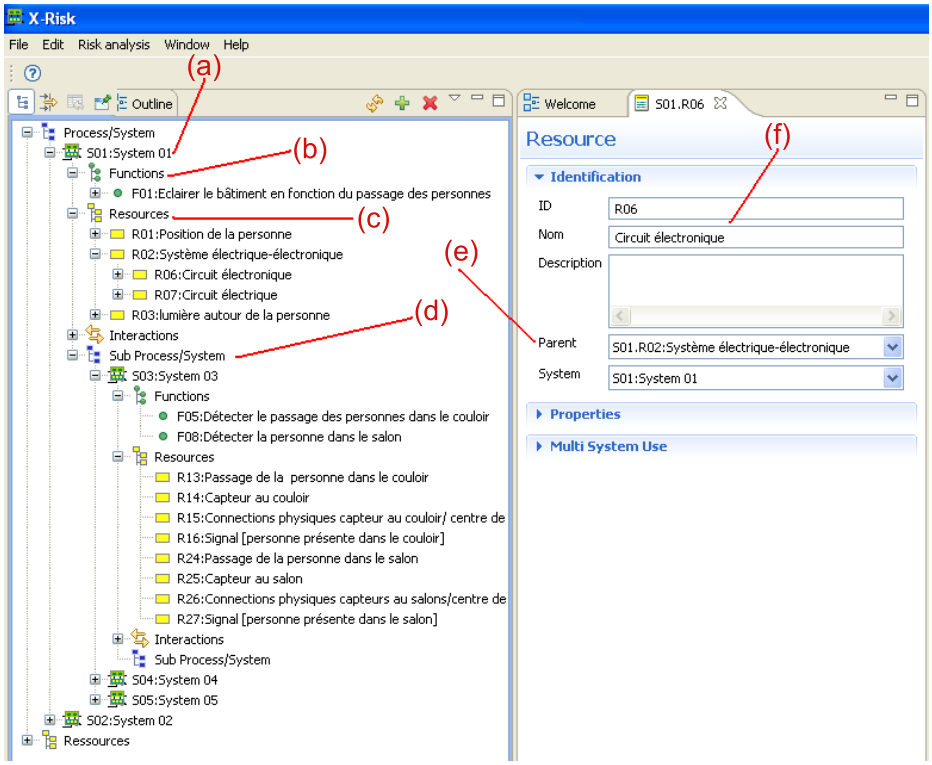


FIG. 3.9 – Interface pour construire le modèle dans FIS

3.5 Conclusion

Dans ce chapitre, nous avons présenté un processus de diagnostic avec interactions homme-automate qui est basé sur le modèle structuro-fonctionnel FIS. Ceci permet de résoudre les difficultés qui se rapportent à la complexité du système. Tout d'abord, le contexte du problème est introduit. Pour aider le travail des services de maintenance, et pour appréhender la variété des systèmes, une démarche de diagnostic qui ne requiert pas une formulation détaillée et complète du système au préalable a été proposée.

Celle-ci permet de décrire un système à différents niveaux de détail en s'appuyant sur la modélisation structuro-fonctionnelle FIS. Le problème de diagnostic est présenté comme un processus interactif homme-automate, où le modèle du système est découvert et explicité au fur et à mesure de l'avancement du processus de diagnostic.

Cette approche permet de calculer les diagnostics et de guider l'expert dans la décomposition du système. Au niveau de l'algorithme de diagnostic, nous avons proposé dans cette partie une méthode pour calculer des diagnostics à partir du modèle FIS et les symptômes formulés par l'expert à chaque interaction. Nous nous sommes focalisés sur les difficultés liées à la description à différents niveaux d'abstraction du système. Les relations hiérarchiques entre les items sont gérées pour qu'elles enrichissent les modèles mais sans fausser les résultats de diagnostic. Les tests consistants sont intégrés dans le calcul des diagnostics afin d'éviter les

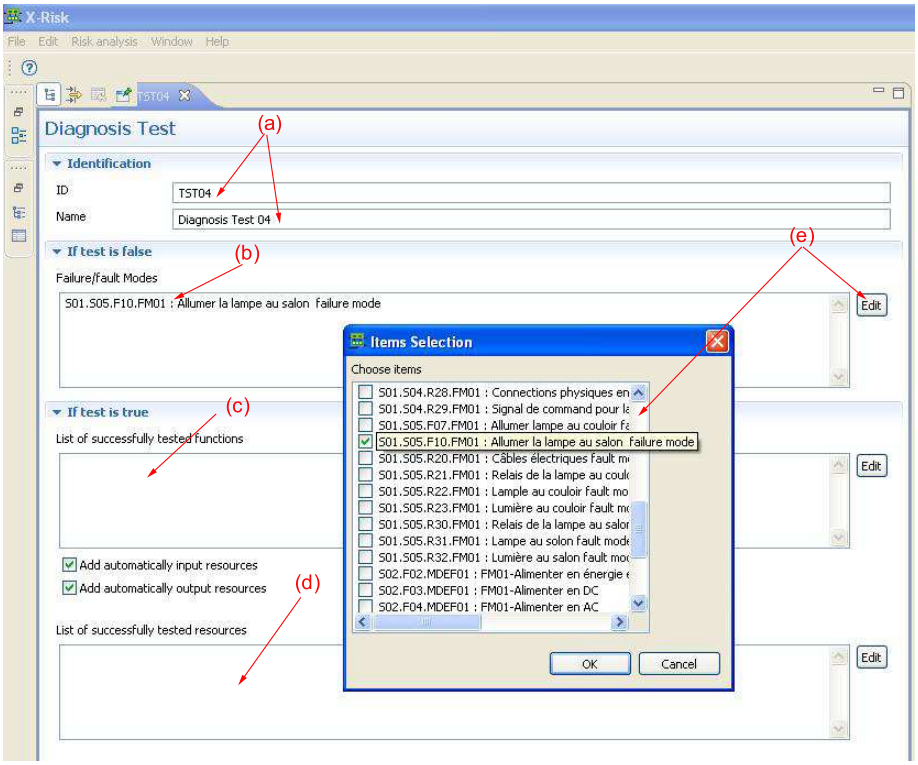


FIG. 3.10 – Interface pour ajouter des test consistants / inconsistants

diagnostics physiquement impossibles.

Contributions en comparaison avec (Ploix et Chazot [2006]) et (Flaus et Pham [2010])

En comparaison avec l'approche structurelle présentée dans (Ploix et Chazot [2006]) (rappelé dans la section 1.4.1), nous avons présenté dans ce chapitre l'approche structuro-fonctionnelle FIS. Cette approche permet à l'expert de décrire un système de façon graphique dans un processus de diagnostic itératif. La description du système est souvent plus facile au niveau fonctionnel qu'au niveau des ressources. FIS permet d'exprimer toutes les relations existant dans un système : abstraction, ressources d'entrée / fonction, fonction / ressources de sortie, etc. Le modèle structuro-fonctionnel permet d'appréhender un grand nombre de systèmes : systèmes industriels, systèmes d'organisation, etc.

Dans (Ploix et Chazot [2006]), seuls les liens de décomposition existent. Nous avons enrichi les possibilités de représentation avec les differents types de lien du modèle FIS : les liens *produit*, *agit* et *est requis par*. Par rapport à (Flaus et Pham [2010]), nous avons détaillé la génération du modèle de dysfonctionnement en utilisant les relations du modèle FIS et nous n'exploitons pas les informations apportées par l'arbre de défaillance dans l'algorithme de résolution. Nous avons intégré la notion de *tests consistants complètement testés* afin d'exploiter au mieux l'expertise pour améliorer les résultats de diagnostic. Ceci n'est pas abordé dans (Ploix et Chazot [2006]).

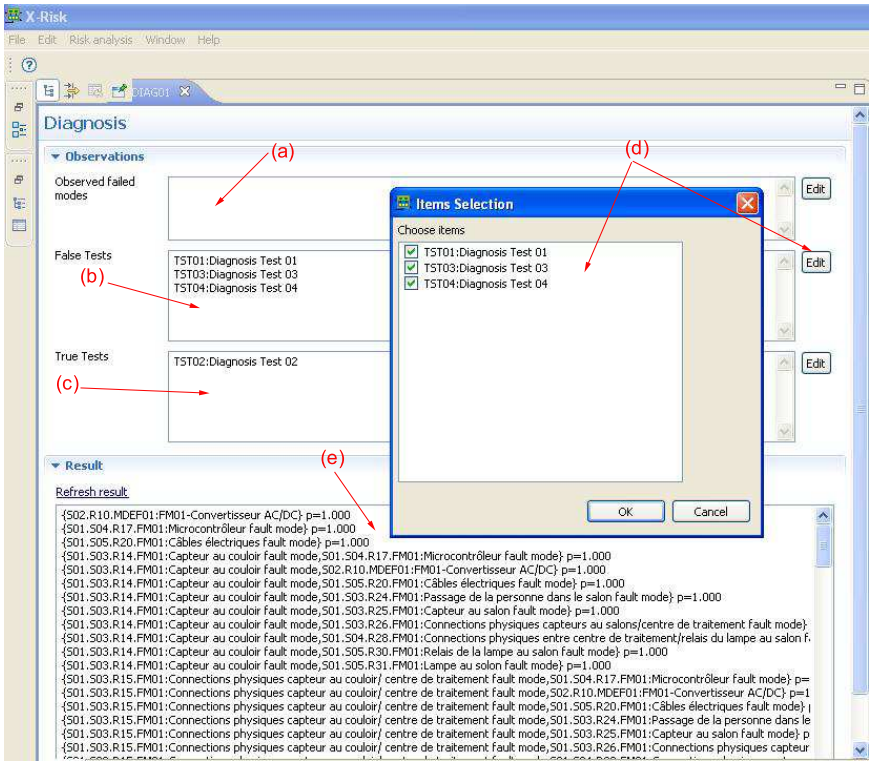


FIG. 3.11 – Interface pour calculer des diagnostics

Chapitre 4

Diagnostic itératif basé sur un modèle comportemental avec propagation de valeurs

4.1 Introduction

Dans cette partie, nous allons étudier le deuxième problème de diagnostic itératif basé sur un modèle comportemental avec propagation de valeurs. Ce problème concerne plus particulièrement les systèmes industriels complexes. En comparaison avec le problème précédent présenté dans le chapitre 3, la complexité du modèle exprimée dans ce problème repose non seulement sur un ensemble important d'éléments impliqués dans le système mais aussi sur le fonctionnement des items exprimés par des contraintes de comportement. Dans ce cas, l'expert a besoin d'un outil d'aide au diagnostic capable de détecter des symptômes. Ceci est la différence principale entre ce problème de diagnostic et celui présenté dans le chapite 3. Pour cette raison, nous proposons dans ce chapitre une autre solution en comparaison avec celle présentée dans le chapitre 3 pour la conception d'un module de diagnostic en s'appuyant sur FIS.

Comme le but est de localiser des pannes dans les systèmes industriels, nous nous limiterons aux ressources matérielles (composants physiques) dans ce problème. Les autres types de ressources (ressources humaines, informationnelles, etc.) qui ne peuvent être modélisées par des modèles de comportement sous forme de relations quantitatives ou qualitatives ne sont pas étudiés ici.

Souvent, pour appréhender la complexité d'un système, le diagnostic est considéré comme un processus où le modèle du système est décrit partiellement et où certaines parties sont affinées au fur et à mesure. En effet, l'analyse diagnostique pour un modèle complet détaillé n'est

pas toujours réaliste parce qu'il peut être trop long et trop couteux à établir. Il n'est d'ailleurs pas nécessaire de modéliser des parties qui n'apportent rien à un raisonnement diagnostique sachant qu'une situation de défaut dans un système complexe est souvent unique. Le diagnostic devient le processus itératif présenté dans la figure 4.1.

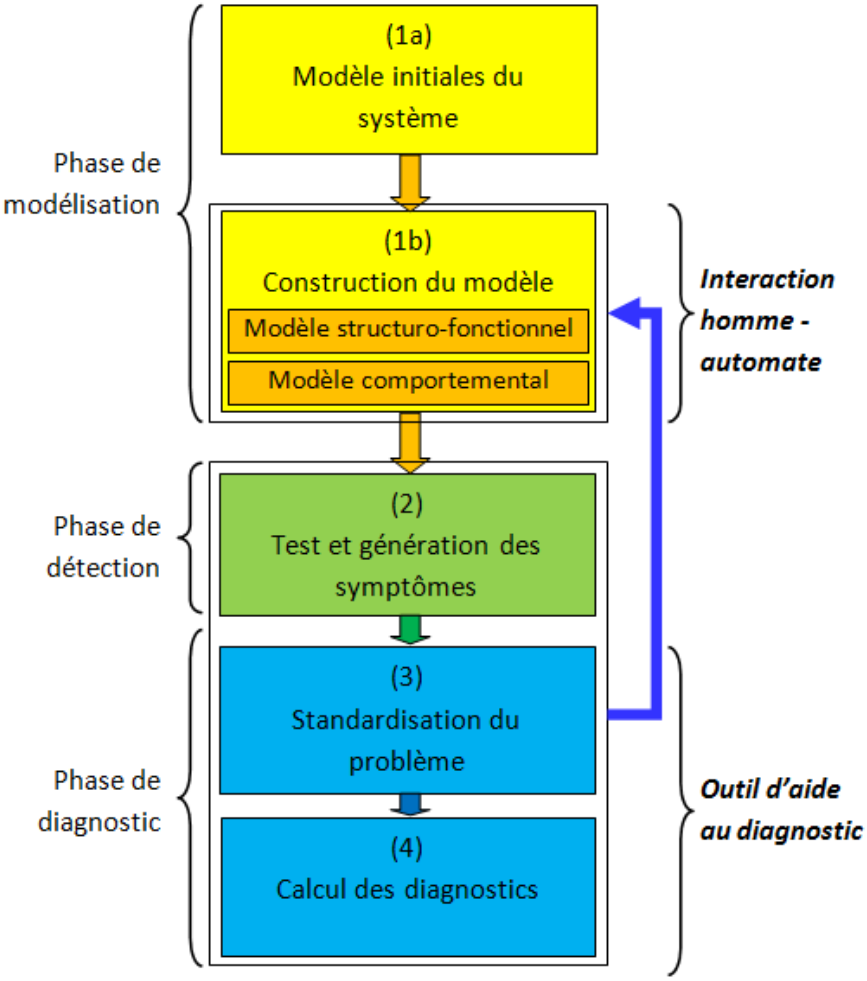


FIG. 4.1 – Diagnostic itératif avec interaction durant la phase de modélisation

La procédure de diagnostic est initiée quand il y a une anomalie qui est détectée grâce à un modèle (le modèle contient les modèles d'observation) initial décrivant partiellement un système avec un jeu d'observations initiales (bloc 1a sur la figure 4.1). Le système d'aide au diagnostic génère automatiquement les symptômes (bloc 2), calcule les diagnostics (blocs 3 et 4). L'expert examine les résultats de diagnostic provisoires, décide de décomposer et d'enrichir le modèle (1b). L'opérateur peut changer l'état du système pour tester et collecter les nouvelles observations (1b). En comparaison avec la figure 3.1, nous voyons que le modèle de comportement est alimenté dans la phase de modélisation comportementale. Le modèle de comportement des items est décrit par des modèles élémentaires. Chaque modèle élémentaire est exprimé par une contrainte de comportement associé avec un mode. La connexion

entre les modèles élémentaires ou entre les items est réalisée par des variables. Le modèle de comportement d'un item est illustré sur la figure 4.2.

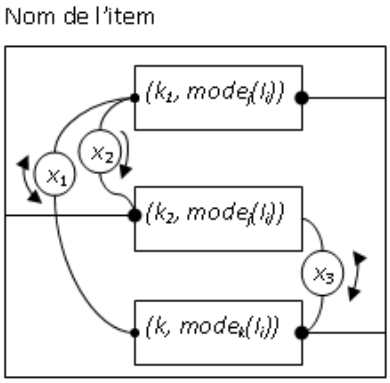


FIG. 4.2 – Modèle de comportement d'un item

En comparaison avec le problème de diagnostic interactif présenté dans le chapitre 3 (figure 3.1), nous voyons que la tâche de détection (bloc 2) est réalisée automatiquement par l'automate dans ce problème. Le calcul des diagnostics à partir des symptômes est identique entre ces deux problèmes de diagnostic. Nous pouvons donc réutiliser les résultats obtenus dans le chapitre 3 sur la tâche de diagnostic (bloc 3 et 4). Au niveau de la résolution, le problème principal auquel nous nous intéressons dans ce chapitre est la génération des symptômes pour un modèle décrit à différents niveaux d'abstraction (bloc 2).

Dans ce chapitre, nous allons tout d'abord préciser, dans la section 4.2, les principes généraux de la résolution de ce problème. Ces principes reposent sur la méthode de modélisation, sur les éléments impliqués dans la formulation d'un problème énoncé qui est formulé par l'expert à chaque itération (blocs 1a et 1b sur la figure 4.1). Ensuite, un outil d'aide au diagnostic est présenté dans la partie 4.3. Pour la conception de cet outil, un nouveau problème de génération de test et de détection de symptômes pour un modèle décrit à différents niveaux d'abstraction est présenté (bloc 2, figure 4.1). Les blocs 3 et 4 sont pareils à ceux présentés dans le chapitre 3 (figure 3.1, page 67). Un exemple est présenté dans la section 4.4 pour illustrer un processus de diagnostic avec interaction durant la phase de modélisation.

4.2 Principe général

Dans cette partie, nous allons résumer rapidement la méthode de modélisation utilisée, la capitalisation des connaissances par l'expert avant de formuler le problème de diagnostic à chaque itération dans le processus. Puis, nous présentons le principe de résolution.

4.2.1 Méthode de modélisation

Pour modéliser le système, nous nous appuyons sur le modèle FIS présenté dans les sections 2.3 et 2.4. Pour détecter automatiquement des symptômes, le modèle de comportement est ajouté pour les items (fonctions ou ressources). Le modèle de comportement n'est souvent disponible que pour certaines ressources. Les autres ressources sont dites non-modélisées.

La connexion entre les modèles élémentaires modélisant des items est réalisée via des variables impliquées dans les contraintes. Ceci permet de réduire certains types de lien qui existent dans le module de diagnostique évoqué dans le chapitre 3. Nous considérons en détail deux types : *fonction/ressources de sortie*, *ressources d'entrée / fonction*.

– Liens *ressources d'entrées / fonction*
Dans le modèle FIS original (conçu pour l'analyse de risques), les types de ressources d'entrée d'une fonction ne sont pas distingués. Afin de gérer la relation hiérarchique entre les items dans notre problème de diagnostic, nous allons distinguer ici deux types de ressources d'entrée d'une fonction. En comparaison avec le modèle FIS présenté sur la figure (2.17), les relations *ressource d'entrées / fonction* représentées par le lien (2) peuvent être détaillées en deux types de liens (2a) et (2b) sur la figure 4.3. (2a) représente des ressources matérielles qui sont utilisées par les fonctions mais ne sont pas consommées ou transformées par les fonctions, par exemple : les machines électriques, les composants électriques, etc. Ce sont des ressources dont les modes (dans FIS original) sont les d-modes dans le module de diagnostic. (2b) représente les ressources d'entrée qui sont consommées ou transformées par les fonctions, par exemple : le flux d'eau, le courant ou la tension électrique, les informations, etc. Ce sont des ressources dont les modes (dans FIS original) sont des symptômes dans le module de diagnostic. Les liens de type (2b) ne seront pas considérés dans ce problème de diagnostic parce qu'ils peuvent être exprimés par des connexions entre les variables quand le modèle comportemental est introduit.

  En faisant le lien avec la définition de l'abstraction sur les modèles élémentaires (définition 2.2.5, page 40), nous voyons que si la description comportementale est disponible pour les items, l'abstraction sur les modèles élémentaires existe implicitement dans la relation *ressources d'entrée / fonction* pour le type (2b). La propriété 3 est donc aboutie. Elle permet de gérer plus tard les abstractions entre les modèles élémentaires qui peuvent apparaître.

  Propriété 3. Soient $\mathscr{K} = \bigcap_{i=1\ldots n}(mode(R), k_i, K'_i)$ et $\mathscr{K}' = \bigcap_{j=1\ldots n'}(mode'(F), k_j, K'_j)$ deux ensembles de modèles élémentaires décrivant respectivement le mode de comportement de R et F. Si R est une ressource d'entrée de la fonction F dans une relation ressource d'entrée/ fonction (type 2a), $\mathscr{K}$ appartient nécessairement à l'ensemble des sous-modèles élémentaires de $\mathscr{K}'$.

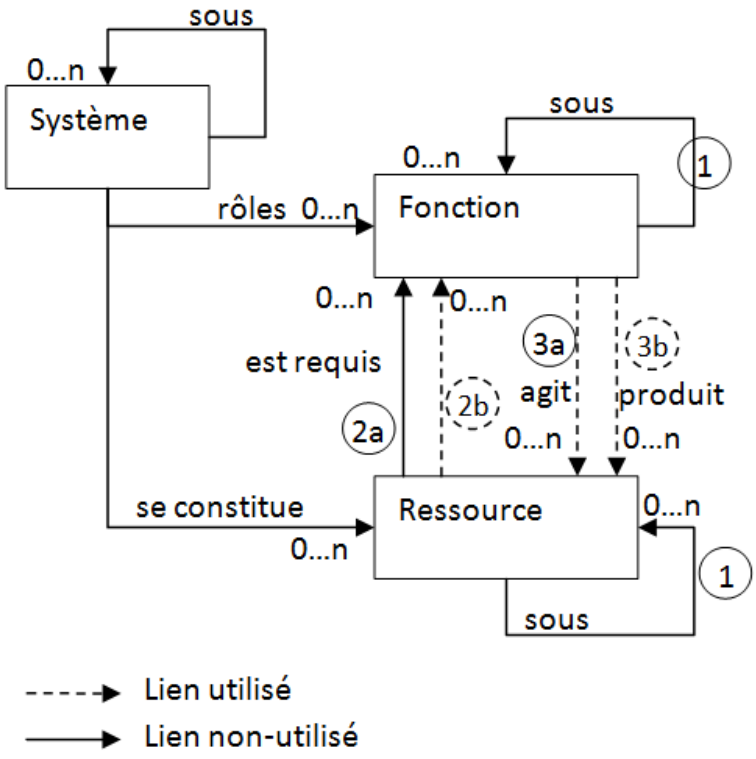


FIG. 4.3 – Présentation du modèle FIS

– Liens *fonction / ressources de sortie*
  Dans la section 2.4.2, nous avons distingué les liens *fonction/ ressources de sortie* en deux types : *agit* et *produit*. Ils sont représentés respectivement par les liens (3a) et (3b) sur la figure 4.3. Les liens (3a) et (3b) ne seront pas considérés dans ce chapitre qu'ils peuvent être exprimés par des connexions entre les variables quand le modèle comportemental est introduit.

En résumé, en comparaison avec le chapitre 3, nous proposons dans ce chapite une autre solution pour faire face aux difficultés liées au comportement du système. Pour construire le modèle du système au fur et à mesure dans le processus de diagnostic, nous utilisons dans ce chapitre le modèle structuro-fonctionnel basé sur FIS et le modèle comportemental. Pour la modélisation structuro-fonctionnelle, nous n'exploitons que les liens d'abstraction (1) et les liens ressources d'entrée/fonction de type (2a) sur la figure 4.3. Le modèle comportementale est ensuite ajouté pour les items (fonctions, ressources). Le modèle comportemental (décrit par des modèles élémentaires) permet de connecter les items via des variables. Les items dont le modèle comportemental n'est pas disponible peuvent être connectés avec les autres items via des relations d'abstraction ou des relations fonction/ressource. Dans l'analyse diagnostique, les liens fonction/ressource et les liens abstraction peuvent être transformés automatiquement en un ensemble de propositions logiques selon le tableau du figure 2.20, page 56.

Notons que les ressources (dans FIS) dont les modes sont les symptômes dans le module de diagnostic (ou les ressources cochées) (voir la section 2.4.1) ne sont plus intéressants à modéliser dans ce chapitre car il peut être exprimé par des variables impliqué dans les contraintes. Dans le modèle, il n'y a que des ressources qui ne sont pas cochées. Pour la solution proposée dans ce chapitre, il n'est donc pas nécessaire de faire apparaître des cases à cocher quand nous passons au module de diagnostic.

Capitalisation des connaissances de l'expert pour la formulation d'un problème de diagnostic

En résumé, à chaque interaction dans le processus de diagnostic, le problème formulé par l'expert comporte :

1. le modèle structuro-fonctionnel construit par l'expert. Dans ce modèle, les informations disponibles sont :
   - une liste d'items (fonction, ressource) avec des modes de défaut spécifiques associés (des modes de défaillances/défauts de chaque fonction et ressources). Notons que les deux modes {ok, cfm} sont ajoutés automatiquement pour chaque item quand il est ajouté dans le système. Le plus souvent, chaque item a seulement ces deux modes {ok, cfm}.
   - des fonctions et des sous-fonctions organisées de façon hiérarchique (chaque fonction peut avoir une liste de fonctions enfants).
   - des ressources et des sous-ressources organisées elles aussi de façon hiérarchique.
   - les modèles de comportement disponibles pour certains items impliqués dans le graphe F/R avec les connexions partielles / complètes entre les modèles élémentaires qui décrivent le comportement des items.
   - des observations *Obs*, alimentées par l'opérateur, qui entre dans les contraintes comportementales et permettent de détecter des symptômes.
   - l'abstraction des modes de défaillances, qui donne pour chaque mode de défaillance/défaut d'une fonction ou ressource, le mode de défaillance ou défaut de la fonction ou ressource parente.
   - un modèle de dysfonctionnement exprimant des relations logiques entre les modes de défaillances des différents items. Ces relations entre les modes des items peuvent être partiellement déduites du graphe structuro-fonctionnel. Des informations complémentaires peuvent être ajoutées pour décrire ces relations. Elles sont exprimées par des propositions logiques. Par exemple, $fm\ (I_1) \rightarrow fm\ (I)$, $fm\ (I_1) \wedge fm\ (I_2) \rightarrow fm\ (I)$, etc...

   Le graphe structuro-fonctionnel peut être transformé et traduit en un ensemble de propositions logiques pour alimenter l'algorithme de diagnostic.
2. de façon optionnelle, la probabilité de défaillance des modes de base (c'est à dire *ok* ou

cfm)

4.2.2 Vues générales sur la résolution du problème

Nous rappelons ici l'ensemble des étapes de résolution du problème représentées sur la figure 4.1. L'étape de modélisation consiste à construire le modèle structuro-fonctionnel et le modèle comportemental du système. L'étape de transformation consiste à transformer le modèle FIS donné par l'expert en un ensemble de propositions logiques sur des modes pour les exploiter dans l'analyse diagnostique. L'étape de calcul des diagnostics permet de chercher tous les diagnostics possibles à partir des symptômes. Ces deux dernières étapes sont réalisées suivant les mêmes principes que ceux présentés dans le chapitre 3. Dans cette partie, nous nous concentrons sur l'étape de test et de génération des symptômes présentés sur la figure 4.1. Cette étape peut être détaillée en deux sous-étapes : la conception des tests de détection en déterminant les sous-systèmes testables (SST) (Blanke *et al.* [2006]; Travé-Massuyès *et al.* [2006]; Yassine [2008]) et la génération des symptômes. Dans notre problème, la description du système est construite au fur et à mesure du processus diagnostic à l'aide d'une approche structuro-fonctionnelle. De nouveaux problèmes apparaissent alors :

– Il faut prendre en compte les niveaux d'abstraction entre modèles dans la recherche des tests. Chaque test est calculé à partir d'un sous-système testable (SST). La recherche des SSTs (Yassine [2008]; Ploix *et al.* [2010]) est rappelée dans la section 4.3.1. Les SSTs générés à partir des modèles combinant différents niveaux d'abstraction peuvent conduire à des tests de détection erronés ou des tests qui peuvent être déduits l'un de l'autre. Par exemple, si le test T_j peut être déduit du test T_i et si T_i est consistant (ou inconsistant), il permet de conclure directement que T_j est consistant (ou inconsistant). Il est nécessaire d'enlever ces tests car ils perturbent les résultats de diagnostic et conduisent à des mesures de coïncidence erronées.
– Il faut prendre en compte l'évolution du système au cours du processus de diagnostic. En effet, à un moment donné, l'opérateur peut changer l'état du système pour le tester. Les observations peuvent donc changer au cours du processus de diagnostic.

Interaction homme-automate pour déterminer les tests consistants complètement testés

Dans la réalité physique, beaucoup de symptômes sont complètement testés (définition 2.1.8, page 35). Par exemple, considérons le test : on met le contacteur en position ON, la lampe s'allume. Ce test permet de dire que les éléments impliqués dans le SST correspondant sont apparemment en mode de fonctionnement normal. Comme les SSTs sont calculés par l'automate, la connaissance sur *les tests complètement testés* ne sera pas fournie au début. La prise en compte du résultat *des tests complètement testés* permettra de réduire le nombre de diagnostics par l'exonération. Les interactions entre l'opérateur et l'automate se produiront

aussi à chaque fois qu'un test consistant est détecté pour vérifier *les tests consistants complètement testés*.

Accumulation de symptôme à chaque interaction

Dans un problème de diagnostic où le modèle du système est construit au fur et à mesure, les symptômes sont collectés automatiquement à partir des observations et du modèle formulé à chaque itération. Un problème pouvant apparaître est l'évolution du système et des observations sur le système. Par exemple, l'expert change la position d'un commutateur électrique pour tester. Dans ce cas, les nouveaux symptômes sont collectés par l'automate à cette étape. Pourtant, certains anciens symptômes ne sont plus détectés à cette interaction. Pour faire face à cette difficulté, une solution proposée est d'accumuler les symptômes à chaque interaction car on considère qu'un système ne peut s'auto-répare : même si un symptôme n'apparaît plus le défaut qui l'a fait apparaître ne peut avoir disparu. Cela veut dire que l'ensemble de symptômes obtenus à une étapes k comporte des nouveaux symptômes détectés à l'étape k et des anciens symptômes collectés durant des étapes précédentes {1,..., k-1}.

Sans hypothèse d'exonération implicite, le nouveau diagnostic obtenu à l'étape k n'est jamais contradictoire avec celui de l'étape précédente : il ne fera que le préciser. Ceci peut être expliqué par *le principe de monotonie*. En effet, soit $P = \bigwedge_{T_i \in \mathscr{T}} Expl(\top(T_i))$ (voir la proposition 3.3.4, page 75) un problème de diagnostic transformé ayant obtenu à l'étape $k-1$, qui conduit aux diagnostics D. A l'étape k, certains nouveaux symptômes sont collectés. Notons T_i' un de ces nouveaux tests. Le principe de monotonie dit que :

$$(P \models D) \rightarrow \big(P \wedge Expl(\top(T_i')) \models D\big)$$

Quand les nouveaux symptômes sont collectés, les diagnostics sont donc plus fins et plus précisés.

4.3 Génération des sous-systèmes testables (SST) en prenant en compte les différents niveaux d'abstraction

Les algorithmes de résolution de CSP (Constraint Satisfaction Problem) nous semblent adaptés à la détection automatique des symptômes à partir d'un ensemble de contraintes décrivant le comportement d'un système. Formellement, un problème de satisfaction de contraintes est défini par un triplet $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ avec $\mathscr{C} = \{k_1, \ldots, k_n\}$ est un ensemble fini de contraintes impliquées dans un système (ou sous-système) à tester. $\mathscr{X} = var(\mathscr{C}) = \{x_1, \ldots, x_n\}$ un ensemble fini de variables impliqué dans $\mathscr{C}$. $\mathscr{D} = \{dom(x_1), \ldots, dom(x_n)\}$ sont les domaines de valeurs qui correspondent aux variables $\{x_1, \ldots, x_n\}$. Un *CSP* consiste à chercher tous les jeux de valeurs correspondant à $\{dom(x_1), \ldots, dom(x_n)\}$ qui satisfont un ensemble de contraintes

donné $\mathscr{C}$. On dit qu'un jeu de valeurs $\{v_1 \in dom(x_1), \ldots, v_n \in dom(x_n)\}$ est une solution de $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ s'il satisfait chaque contrainte $(C) \in \mathscr{C}$. Or l'ensemble de toutes les contraintes disponibles d'un système n'est pas un CSP pertinent :

- Un *CSP* $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ n'a pas de solution s'il existe en même temps deux contraintes qui décrivent deux modes différents d'un même item dans l'ensemble de contraintes $\mathscr{C}$. En fait, il est impossible d'avoir deux modes d'un même item en même temps. Il est donc nécessaire de prédéterminer $\mathscr{C}$ tel qu'il n'existe pas dans $\mathscr{C}$ deux contraintes qui décrivent deux modes différents d'un même item.
- Dans le contexte du diagnostic interactif, le modèle du système est construit itérativement. Des abstractions entre les modèles construits apparaissent. Il est difficile d'adapter un CSP à ce contexte.
- L'application d'algorithmes CSP au système complet ne permet pas de localiser les défauts au niveau des sous-systèmes. Soit un *CSP* $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ avec $\mathscr{C}$ l'ensemble de toutes les contraintes disponibles décrivant le système. Si aucune solution n'est trouvée pour ce *CSP*, cela veut dire qu'il y a un défaut au niveau des items décrit par $\mathscr{C}$ mais il n'est pas possible d'affiner ce diagnostic.

Pour ces raisons, nous proposons d'associer la méthode structurelle proposée dans (Ploix *et al.* [2009, 2010]) à l'approche CSP pour chercher tout d'abord les sous-systèmes testables (SST). Chaque SST est un ensemble de modèles élémentaires qui peut engendrer un test de détection. Ensuite, la propagation de valeurs détermine des jeux de valeurs correspondant à chaque SST et vérifie la satisfaction des contraintes à partir des observations (figure 4.4). Comme chaque contrainte correspond à un modèle élémentaire dans la méthode structurelle, il est facile d'éviter l'existence de deux contraintes qui décrivent deux modes différents d'un item dans un SST trouvé.

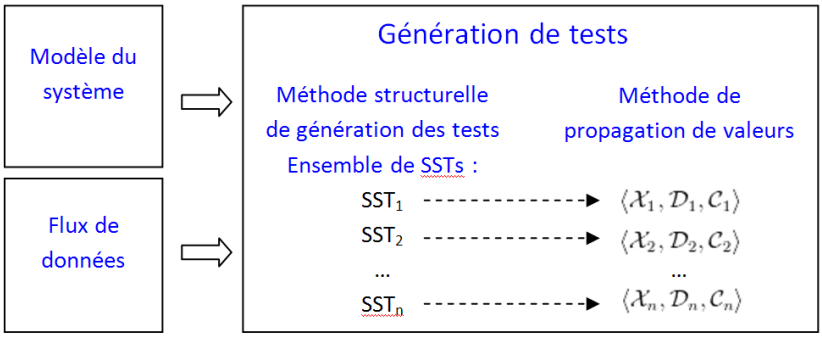


FIG. 4.4 – Procédure de génération des tests

La procédure de détection des symptômes est divisée en deux étapes : *la recherche des SSTs qui permet de prendre en compte les différents niveaux d'abstraction du modèle* et *la propagation de valeurs au sein de chaque SST pour détecter les symptômes*.

Dans la suite, nous allons étudier le problème de génération des SSTs en prenant en compte plusieurs niveaux d'abstraction du modèle. Nous calculons tout d'abord les SSTs par la méthode structurelle. Puis nous allons montrer comment les SSTs sont gérés aux différents niveaux d'abstraction. La méthode de propagation de valeurs est utilisée ensuite sur chaque *SST* pour vérifier si les tests sont inconsistants ou consistants. La méthode de propagation de valeurs est reformulée en se basant sur le problème de *satisfaction de contraintes (CSP)* en s'appuyant sur la méthode de *vérification avant (forward checking)*. En plus de la méthode originale, nous avons proposé des améliorations qui permettent de prendre en compte la notion de déductibilité des variables (Yassine [2008]) dans la propagation de valeurs. Comme la partie de propagation de valeurs n'est pas une contribution principale de ce travail, elle peut être consultée dans l'annexe C.

4.3.1 Recherche des SST par la méthode structurelle

Dans cette partie, l'approche structurelle pour la génération des sous-systèmes testables est présentée. Cette méthode a été présentée dans (Ploix *et al.* [2005b]) et a été améliorée par (Yassine [2008]), (Ploix *et al.* [2009, 2010]). Commençons par le rappel de la définition de *l'ensemble de contraintes solution* (Yassine [2008]).

Définition 4.3.1. *(Ensemble de contraintes solution) Supposons que K soit un ensemble de contraintes, $var(K)$ est l'ensemble des variables apparaissant dans K et v une variable dans $var(K)$ caractérisée par son domaine $dom(v)$. K est un ensemble de contraintes solution pour v si en utilisant K, il est possible de trouver un ensemble de valeurs S pour v tel que $S \subset dom(v)$. Un ensemble de contraintes solution pour v est minimal si il n'existe pas de sous-ensemble de K qui soit également un ensemble de contraintes solution pour v. Un ensemble minimal de contraintes solution K pour v est noté : $K \vdash v$.*

La définition de *sous-système testable (SST)* est introduit.

Définition 4.3.2. *(sous-système testable) Supposons que K soit un ensemble de contraintes. K est testable si et seulement si il existe deux sous ensembles distincts $K_1 \subset K$, $K_2 \subset K$ tels que $K_1 \nsubseteq K_2$ et $K_2 \nsubseteq K_1$, et une variable $v \in var(K)$ tel que $K_1 \vdash v$ et $K_2 \vdash v$. Si cette propriété est satisfaite, il est en effet possible de vérifier si l'ensemble de valeurs S_1 déduit de K_1 est consistant avec l'ensemble de valeurs S_2 déduit de K_2 : $S_1 \cap S_2 \neq \emptyset$.*

Par extension, un sous-système testable est considéré dans ce travail comme un ensemble de modèles élémentaires (section 2.1.4, page 30).

L'approche structurelle pour la génération des sous-systèmes testables permet de remédier à des limites soit en terme de complexité de calcul, soit en en terme de classe de systèmes adressables. Elle permet de tenir compte de la déductibilité (Yassine [2008]) des variables par

rapport aux contraintes. L'idée principale de cette approche est d'utiliser *l'opérateur jointure* venant de l'algèbre relationnel pour propager entre les contraintes (appelées aussi structure) via les variables communes. La propagation de variables (et non de valeurs) permet de chercher tous les SSTs possibles. Cette approche est intégrée dans le logiciel DXLab développé au laboratoire G-SCOP.

Exemple 14. Prenons ici l'exemple simple d'un système d'éclairage dans le bâtiment. Dans cet exemple, nous allons considérer la description du modèle du système disponible lors d'une itération (figures 4.5 et 4.6) dans le processus de diagnostic pour voir comment les SSTs sont générés aux différent niveaux d'abstraction du modèle. Cet exemple sera détaillé plus tard avec plusieurs étapes d'interaction dans la partie application pour illustrer comment un processus de diagnostic avec interaction durant la phase de modélisation se manifeste. Pour marquer la différence avec le problème précédent avec interaction durant la phase de test, cet exemple est présenté avec l'hypothèse que l'expert ne peut pas déterminer si un test est vrai ou faux. Il a besoin de l'outil d'aide au diagnostic pour générer les symptômes.

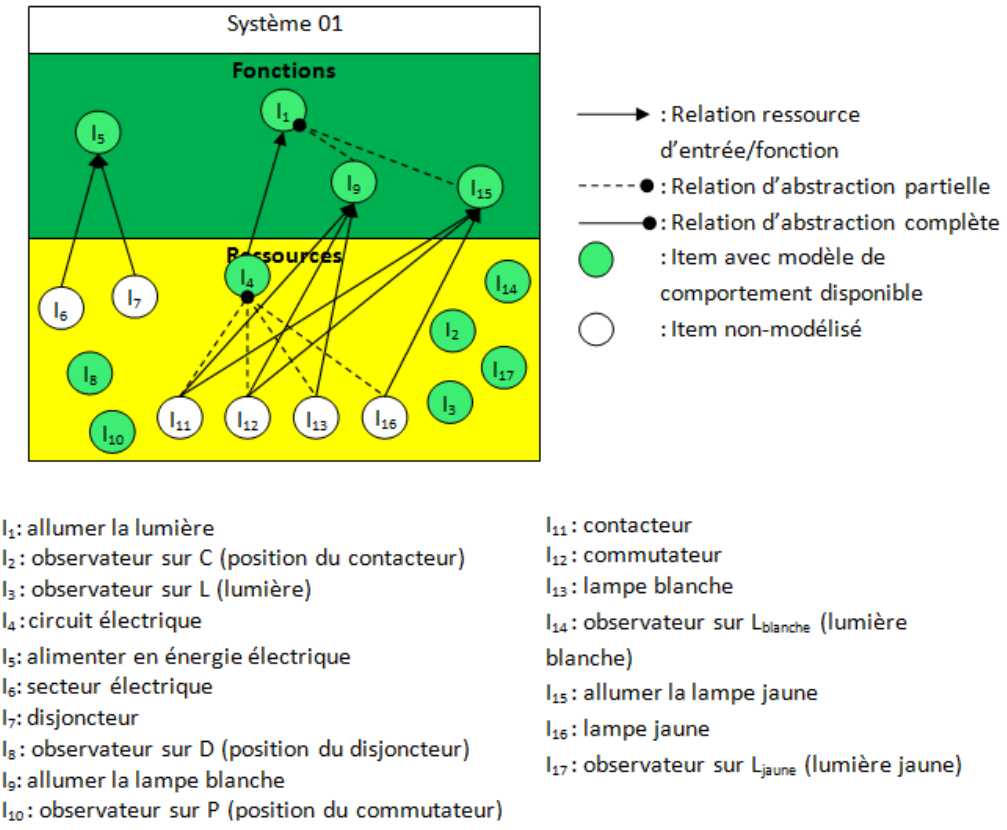


FIG. 4.5 – Modélisation FIS

Tout commence lorsqu'un client informe l'outil d'aide au diagnostic que le système d'éclairage dans le bâtiment est défectueux. L'expert commence le processus de diagnostic pour localiser les défauts. Il commence par construire le modèle du système. Au fur et à mesure, le modèle du système est construit à différents niveaux d'abstraction (4.5).

– Il ajoute la fonction I_1 (*allumer la lumière*). Le modèle de comportement de I_1 est décrit par le modèle élémentaire $(ok(I_1), k_1)$. k_1 est représenté par le tableau 4.1. Il ajoute des observateurs I_2 et I_3 qui sont décrits respectivement par des modèles élémentaires $(ok(I_2), k_2 : C = \widetilde{C})$ et $(ok(I_3), k_3 : L = \widetilde{L})$ avec C est la position d'un contacteur, *dom(C)={on, off}*, L est la lumière, *dom(L)={présent, absent}*.

Modes	V (tension)	C (position du contacteur)	L (lumière)
ok	présent	on	présent

TAB. 4.1 – Contrainte k_1 (fonction allumer la lumière)

– Il détaille la fonction I_1 en intégrant une tension électrique . Une ressource d'entrée de I_1 est I_4 (*le circuit électrique*). Le modèle de comportement de I_4 est décrit par le modèle élémentaire $(ok(I_4), k_4)$ avec k_4 représenté par le tableau 4.2. V est la tension avec *dom(V)={présent, absent}*.

Modes	V (tension)	C (position du C contacteur)	$L(lumire)$ $L(lumire)$
ok	présent	off	absent
ok	présent	on	présent
ok	absent	off	absent
ok	absent	on	absent

TAB. 4.2 – Contrainte k_4 (le circuit électrique)

– Il enrichit le modèle par la fonction I_5 (*alimenter en énergie électrique*). I_5 est décrit par le modèle élémentaire $(ok(I_5), k_5)$ avec k_5 représenté par le tableau 4.3. La variable D est la position du disjoncteur avec *dom(D)={fermé, ouvert}*. I_6 (*le secteur électrique* et I_7 (*le disjoncteur*) sont deux ressources d'entrée de la fonction I_5. I_8 est un observateur sur la position du *disjoncteur*. I_8 est décrit par $(ok(I_8), k_8 : \ D = \widetilde{D})$.

Modes	D (position du disjoncteur)	V (tension)
ok	fermé	présent
ok	ouvert	absent

TAB. 4.3 – Contrainte k_5 (fonction alimenter en énergie électrique

– L'expert décompose et enrichit le modèle en décomposant la fonction I_1 (*allumer la lumière*) en la sous-fonction I_9 (*allumer la lumière blanche*) pour tester. I_9 est décrit par le modèle élémentaire $(ok(I_9), k_9)$. k_9 est représenté par le tableau 4.4. P est la position du commutateur avec $dom(P) = \{x_{blanc}, x_{jaune}\}$. I_{11} ($contacteur$), I_{12} ($commutateur$) et I_{13}

(*lampe blanche*) sont les ressources d'entrée de la fonction I_9. I_{10} est un observateur sur la position du commutateur. I_{10} est décrit par $(ok(I_{10}), k_{10} :\ P = \widetilde{P})$. I_{14} est un observateur sur la position du commutateur. I_{14} est décrit par $(ok(I_{14}), k_{14} :\ L_{blanc} = \widetilde{L_{blanc}})$.

Modes	V	C	P	L_{blanc}
ok	*présent*	on	x_{blanc}	*présent*

TAB. 4.4 – Contrainte k_9 (fonction allumer la lumière blanche)

– L'expert décompose et enrichit le modèle en décomposant la fonction I_1 (*allumer la lumière*) en autre sous-fonction I_{15} (*allumer la lumière jaune*) pour tester. I_{15} est décrit par le modèle élémentaire $(ok(I_{15}), k_{15})$. k_{15} est représenté par le tableau 4.5. I_{11} (*contacteur*), I_{12} (*commutateur*) et I_{16} (*lampe jaune* sont les ressources d'entrée de la fonction I_{15}. I_{17} est décrit par $(ok(I_{17}), k_{17} :\ L_{jaune} = \widetilde{L_{jaune}})$.

Modes	V (tension)	C (position du contacteur)	P (position du commutateur)	L_{jaune} (lumière)
ok	*présent*	on	x_{jaune}	*présent*

TAB. 4.5 – Contrainte k_{15} (fonction allumer la lumière jaune)

Afin de chercher les SSTs, la structure des contraintes est fournie à DXLab (voir annexe B pour le détail sur la méthode structurelle de génération des SSTs). La structure des contraintes et les variables sont présentées sur la figure 4.6. Dans cette figure, chaque boîte rectangulaire représente un modèle élémentaire qui comporte une contrainte décrivant un mode. Chaque cercle représente une grandeur physique. Les variables impliquées dans les modèles élémentaires se connectent via des variables d'une même grandeur physique. Ces connexions peuvent être partielles ou complètes (voir la section 2.1.4).

La recherche des SSTs ne s'appuie que sur les items avec des modèles de comportement disponibles. L'ensemble des items avec des modèles de comportement disponibles sont $\{I_1, I_2, I_3, I_4, I_5, I_8, I_9, I_{10}, I_{11}, I_{14}, I_{15}, I_{17}\}$ (figure 4.5). Ils sont fournis à DXLab par :

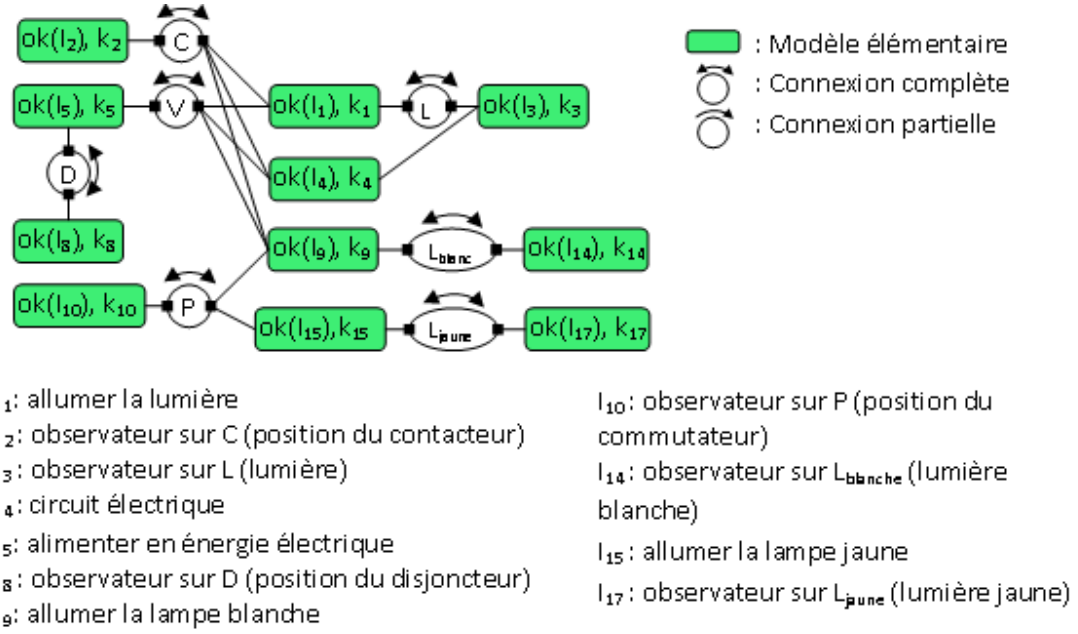


FIG. 4.6 – Modèle de comportement du système

$$
\begin{aligned}
k_{01} &= |\{-\},\{L,V,C\}|\ \ models\ \ I_1\\
k_{02} &= |\{-\},\{C\}|\ \ models\ \ I_2\\
k_{03} &= |\{-\},\{L\}|\ \ models\ \ I_3\\
k_{04} &= |\{-\},\{L,V,C\}|\ \ models\ \ I_4\\
k_{05} &= |\{-\},\{V,D\}|\ \ models\ \ I_5\\
k_{08} &= |\{-\},\{D\}|\ \ models\ \ I_8\\
k_{09} &= |\{-\},\{V,P,C,L_{blanc}\}|\ \ models\ \ I_9\\
k_{10} &= |\{-\},\{P\}|\ \ models\ \ I_{10}\\
k_{14} &= |\{-\},\{L_{blanc}\}|\ \ models\ \ I_{14}\\
k_{15} &= |\{-\},\{V,P,L_{jaune},C\}|\ \ models\ \ I_{15}\\
k_{17} &= |\{-\},\{L_{jaune}\}|\ \ models\ \ I_{17}
\end{aligned}
$$

|{variables non-déductives},{variables déductives}| donne la structure d'un modèle élémentaire correspondant à une contrainte. Dans cet exemple, les contraintes sont données sous forme de tables de comportement. Les variables sont considérées comme déductibles. L'ensemble de variables non-déductibles dans chaque structure est vide. Les sous-systèmes testables (SST) minimaux sont trouvés par DXLab.

$SST_{01} : support = k_{15}[I_{15}], k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_2[I_2], k_{17}[I_{17}]$
$Formula = ((((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$
$SST_{02} : support = k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_3[I_3], k_2[I_2], k_1[I_1]$
$Formula = (((k_1 - L - k_3) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$
$SST_{03} : support = k_{15}[I_{15}], k_{10}[I_{10}], k_3[I_3], k_2[I_2], k_1[I_1], k_{17}[I_{17}]$
$Formula = (((k_1 - L - k_3) - C - k_2) - V - (((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2))$
$SST_{04} : support = k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_4[I_4], k_3[I_3], k_2[I_2]$
$Formula = (((k_4 - L - k_3) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$
$SST_{05} : support = k_{15}[I_{15}], k_{10}[I_{10}], k_4[I_4], k_3[I_3], k_2[I_2], k_{17}[I_{17}]$
$Formula = (((k_4 - L - k_3) - C - k_2) - V - (((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2))$
$SST_{06} : support = k_9[I_9], k_{14}[I_{14}], k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2]$
$Formula = ((k_5 - D - k_8) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$
$SST_{07} : support = k_4[I_4], k_3[I_3], k_2[I_2], k_1[I_1]$
$Formula = (((k_4 - L - k_3) - C - k_2) - V - ((k_1 - L - k_3) - C - k_2))$
$SST_{08} : support = k_{15}[I_{15}], k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2], k_{17}[I_{17}]$
$Formula = ((k_5 - D - k_8) - V - (((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2))$
$SST_{09} : support = k_8[I_8], k_5[I_5], k_3[I_3], k_2[I_2], k_1[I_1]$
$Formula = ((k_5 - D - k_8) - V - ((k_1 - L - k_3) - C - k_2))$
$SST_{10} : support = k_8[I_8], k_5[I_5], k_4[I_4], k_3[I_3], k_2[I_2]$
$Formula = ((k_5 - D - k_8) - V - ((k_4 - L - k_3) - C - k_2))$

DXLab donne structurellement un ensemble de sous-systèmes testables. Une *formule* montre comment les structures se lient par les variables. Ces connexions sont données par les définitions (2.1.5) et (2.1.6).

4.3.2 Gestion des SST à différents niveaux d'abstraction

Dans ce problème de diagnostic interactif dans la phase de modélisation, le modèle du système est construit au fur et à mesure. Les SSTs engendrés permettent de mettre en évident de nouveaux symptômes. A cause des relations hiérarchiques entre les modèles élémentaires, la méthode structurelle de génération des SSTs peut conduire à des SSTs inutiles. Ces SSTs sont dits inutiles parce qu'ils ne permettent pas de générer de nouveaux symptômes. Nous allons détailler ensuite pourquoi ces SSTs sont inutiles. Puis, nous proposerons une solution pour gérer ces SSTs dans le contexte du problème de diagnostic interactif dans la phase de modélisation. Il y a deux types de test qui sont dits inutiles :

1. les SSTs qui ne permettent pas de tester le système. Ce sont des $SSTs$ qui contiennent à la fois un modèle élémentaire $\mathscr{K}$ et son modèle abstrait $\mathscr{K}'$ (l'abstraction sur les modèles élémentaires est donnée par la définition 2.2.5). Pour tester le système, il

faut chercher des tests de détection exprimés sous forme de relations de redondance analytique (RRA). La recherche des RRAs s'appuie sur la propagation de valeurs via des variables au sein de chaque SST. Cette propagation exprime un échange de flux d'information entre les items par la connexion entre les modèles élémentaires. La propagation de valeurs d'un modèle élémentaire à son modèle abstrait exprime un échange d'information d'un item à lui même (peut-être via ses sous-items), cela peut faire apparaître structurellement des circuits de propagation. C'est pourquoi un *SST* qui contient à la fois un modèle élémentaire et son modèle élémentaire abstrait ne permet pas de tester le système. Nous prenons l'exemple du sous-système testable SST_{02} trouvé dans l'exemple 14, la structure de SST_{02} est donné par :

$SST_{02} : support = k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_3[I_3], k_2[I_2], k_1[I_1]$
$Formula = (((k_1 - L - k_3) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$

La formule (formula) correspondant à SST_{02} montre que nous pouvons propager des valeurs de $(ok(I_1), k_1)$ à $(ok(I_9), k_9)$ via la variable V. Le RRA obtenu par SST_{02} ne permet donc pas de tester le système parce que I_9 est le sous-item de I_1.
Pour gérer ces SSTs, nous donnons ici la notion d'un *sous-système testable non-fécond*.

Définition 4.3.3. *(Sous-système testable non-fécond) Soit un sous-système testable correspondant à un ensemble de modèles élémentaires $\mathscr{K}$, le SST est dit non-fécond s'il existe deux modèles élémentaires $K_i \in \mathscr{K}$ et $K_j \in \mathscr{K}$ tels que $\{K_i\}$ est l'abstraction de $\{K_j\}$.*

Ainsi, selon cette définition, s'il existe une relation d'abstraction (définition 2.2.5) entre les modèles élémentaires impliqués dans un *SST*, ce *SST* est non-fécond. A partir de l'ensemble des SSTs donnés par la méthode structurelle de génération de SSTs, les *SSTs non-féconds* sont retirés.
En nous appuyant sur la définitions de SST non-fécond et sur les relations entre les items dans le graphe F/R formulées dans la rubrique 2.4, nous donnons ici les règles pour retirer les SSTs non-féconds. Supposons que $\mathscr{K}_i = (mode(I_i), k_i, K'_i)$ et $\mathscr{K}_j = (mode(I_j), k_j, K'_j)$ soient deux ensembles de modèles élémentaires décrivant respectivement le comportement de I_i et I_j.

- **Règle 1 :** Si I_j est un sous-item de I_i dans une relation d'abstraction (sous-fonction/super-fonction, sous-ressource/super-ressource), $\mathscr{K}_j$ appartient nécessairement à l'ensemble des sous-modèles élémentaires de $\mathscr{K}_i$ (ceci est expliqué par la propriété 1, page 41). Un SST comporte à la fois $\mathscr{K}_j$ et $\mathscr{K}_i$ sera retiré parce qu'il est non-fécond selon la définition 4.3.3.
- **Règle 2 :** Si I_j est une ressource d'entrée de la fonction I_i dans une relation de

ressource d'entrée / fonction, $\mathscr{K}_j$ appartient nécessairement à l'ensemble des sous-modèles élémentaires de $\mathscr{K}_i$ (ceci est expliqué par la propriété 3, page 98). Un SST comportant à la fois $\mathscr{K}_j$ et $\mathscr{K}_i$ sera retiré parce qu'il est non-fécond selon la définition 4.3.3.

– **Règle 3 :** S'il existe $\mathbb{R} = \{I_1, \ldots, I_k\}$ un ensemble complet des sous-ressources d'entrée de la fonction I_i et si $\mathbb{R}$ appartient à l'ensemble des sous-ressources de la super-ressource I_j (figure 4.7), alors $\mathscr{K}_i$ appartient nécessairement à l'ensemble des sous-modèles élémentaires de $\mathscr{K}_j$. Un SST comportant à la fois $\mathscr{K}_j$ et $\mathscr{K}_i$ sera donc retiré parce qu'il est non-fécond selon la définition 4.3.3.

Explication du règle 3 : Si $\mathbb{R} = \{I_1, \ldots, I_k\}$ est un ensemble complet des sous-ressources d'entrée de la fonction I_i, cela veut dire que I_i est un rôle (ou une fontion) de l'ensemble des ressources $\{I_1, \ldots, I_k\}$. Comme $\{I_1, \ldots, I_k\}$ est aussi l'ensemble des sous ressources de I_j dans une relation d'abstractiton (partielle ou complète), I_i doit être une fonction (ou un rôle) de I_j (I_j peut avoir plusieurs rôles). Quand le modèle comportemental est ajouté pour I_i et I_j, il existe donc une relation hiérarchique entre les modèles élémentaires décrivant I_i et ceux décrivant I_j. Cette relation n'est pas explicitée dans le graphe F/R.

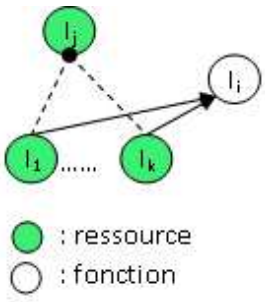


FIG. 4.7 – Modélisation FIS

Exemple 15. Revenons à l'ensemble des SSTs donnés par DXLab dans l'exemple 14. Les sous-systèmes testables SST_{02}, SST_{03}, SST_{07}, SST_{04}, SST_{05} sont retirés.
Pour le SST_{02}, nous avons $\{ok(I_1), k_1\} \in SST_{02}$ et $\{ok(I_9), k_9\} \in SST_{02}$. Comme I_1 est l'abstraction de I_9, $\{ok(I_1), k_1\}$ est l'abstraction de $\{ok(I_9), k_9\}$ selon la propriété 1 (page 41). SST_{02} est non-fécond selon la définition 4.3.3 parce qu'il comporte deux modèles élémentaires dont l'un est l'abstraction de l'autre. SST_{02} est donc retiré du fauit de la règle 1. De la même manière, SST_{03} est retiré parce que $\{ok(I_1), k_1\} \in SST_{03}$, $\{ok(I_{15}), k_{15}\} \in SST_{03}$ et I_1 est l'abstraction de I_{15}.
Pour le SST_{07}, nous avons $\{ok(I_1), k_1\} \in SST_{07}$ et $\{ok(I_4), k_4\} \in SST_{07}$. Comme I_4 est une ressource d'entrée de I_1, $\{ok(I_1), k_1\}$ est l'abstraction de $\{ok(I_4), k_4\}$ d'après la propriété 3 (page 98). SST_{02} est non-fécond d'après la définition 4.3.3 parce qu'il comporte deux modèles élémentaires dont l'un est l'abstraction de l'autre. SST_{07} est donc retiré conformément à la règle 2.

Concernant le SST_{04}, nous avons $(ok(I_4),k_4) \in SST_{04}$ et $(ok(I_9),k_9) \in SST_{04}$. Comme $\{I_{11},I_{12},I_{13}\}$ est un ensemble complet des ressources d'entrée de la fonction I_9, et $\{I_{11},I_{12},I_{13}\}$ sont des sous-items de I_4 alors $(ok(I_4),k_4)$ doit être l'abstraction de $(ok(I_9),k_9)$ comme nous avons expliqué dans la règle 3. SST_{04} est donc retiré. De la même manière, SST_{05} est retiré parce que $\{ok(I_4),k_4\} \in SST_{05}$, $\{ok(I_{15}),k_{15}\} \in SST_{05}$, et $\{I_{11},I_{12},I_{16}\}$ est à la fois un ensemble complet des ressources d'entrée de la fonction I_{15} et des sous-items de I_4.

2. les SSTs qui sont déductibles des autres. L'expression *SST_i est déductible de SST_j* signifie que si le symptôme correspondant à SST_j est connu, il est possible de déduire le symptôme correspondant à SST_i à l'aide des relations hiérarchiques (super-item/sub-item). Souvent, les modèles élémentaires impliqués dans SST_i sont plus abstraits que ceux impliqués dans SST_j. Ce phénomène peut être expliqué grâce à la définition de l'abstraction sur les modèles élémentaires (définition 2.2.5, page 40).

   Prenons un exemple trivial de trois $SSTs$. Chaque SST comporte des modèles élémentaires :

$$SST_1 = \{(ok(I_1),k_1 : x*y=6),(ok(I_4),k_4 : x=\widetilde{x}),(ok(I_5),k_5 : y=\widetilde{y})\}$$
$$SST_2 = \{(ok(I_2),k_2 : z-y-1=0),(ok(I_5),k_5 : y=\widetilde{y}),(ok(I_6),k_6 : z=\widetilde{z})\}$$
$$SST_3 = \{(ok(I_3),k_3 : x>0 \rightarrow z>0),(ok(I_4),k_4 : x=\widetilde{x}),(ok(I_6),k_6 : z=\widetilde{z})\}$$

   avec k_1, k_2, k_3 sont des contraintes de comportement, et k_4, k_5, k_6 sont des contraintes terminales décrivant le modèle des observateurs (définition 2.1.3, page 27).

   Supposons que I_3 soit le super-item de I_1 et I_2, avec $\{(ok(I_3),k_3 : x>0 \rightarrow z>0)\}$ l'abstraction sur les modèles élémentaires de $\{(ok(I_1),k_1 : x*y=6),(ok(I_2),k_2 : z-y-1=0)\}$. Ceci satisfait la condition que pour tous les tuples de valeurs (x,y,z), si k_1 et k_2 sont satisfaits, k_3 est nécessairement satisfait (définition 2.2.5, page 40). Supposons que I_1 ne soit pas en mode de bon fonctionnement. Il existe toujours une observation, par exemple, $(x,y,z) = (3,-2,-1)$, tel que k_1 est insatisfait et k_3 est insatisfait. Il y a donc un défaut dans le sous-système testable SST_3. Sans compensation des défauts, si T_1 (correspondant à SST_1) est inconsistant alors T_3 (correspondant à SST_3) est inconsistant, ou T_3 est déductible de T_1.

   Disons qu'il existe un défaut dans un SST_i s'il existe une observation tel que une contrainte k_j impliqué dans SST_i est insatisfait. En revanche, si toutes les contraintes impliquées dans SST_i sont satisfaites par apport à une observation, il n'est pas sure que tous les items impliqué dans SST_i sont en bon fonctionnement car nous n'avons pas testé avec toutes les observations possibles. Pour cette raison, nous proposons ensuite la formulation de la déductibilité des SST en s'appuyant sur les modes mais pas sur les contraintes. De plus, les symptômes sont en fin formulés en forme de disjonction de modes

Définition 4.3.4. *(SST_i est déductible de SST_j) Soient deux sous-systèmes testables SST_i et SST_j. SST_i est déductible de SST_j s'il existe un défaut dans SST_j alors il existe un défaut dans SST_i*

Exemple 16. Supposons que nous ayons deux sous-systèmes testables $SST_j = \{(mode(I_A),k_A),(mode(I_B),k_B),(mode(I_C),k_C),(mode(I_D),k_D)\}$ et $SST_i = \{(mode(I_1),k_1),(mode(I_2),k_2),(mode(I_3),k_3)\}$ qui correspondent respectivement à deux tests T_j et T_i. Si T_j et T_i sont insatisfaits pour un jeu d'observations, les deux explications suivant peuvent être déduites : $E_j = Expl(\top(T_j)) = (\neg mode(I_A) \vee \neg mode(I_B) \vee \neg mode(I_C) \vee \neg mode(I_D))$ et $E_i = Expl(\top(T_i)) = (\neg mode(I_1) \vee \neg mode(I_2) \vee \neg mode(I_n))$. Les relations hiérarchiques (item enfant/item parent) sont représentées par des arcs (figure 2.6). Considérons trois cas :

– Cas 1 (figure 4.8) : Selon la proposition 2.2.14 (40), les relations hiérarchiques présentées sur la figure 4.8 donnent $\neg mode(I_A) \rightarrow \neg mode(I_1)$, $\neg mode(I_B) \rightarrow \neg mode(I_2)$, $\neg mode(I_C) \rightarrow \neg mode(I_2)$, $\neg mode(I_D) \rightarrow \neg mode(I_3)$. Nous avons donc $E_j \rightarrow E_i$. Selon la définition 4.3.4, SST_i est déductible de SST_j.
– Cas 2 (figure 4.9) : De la même manière, SST_i est déductible de SST_j.
– Cas 3 (figure 4.10) : SST_i n'est pas déductible de SST_j.

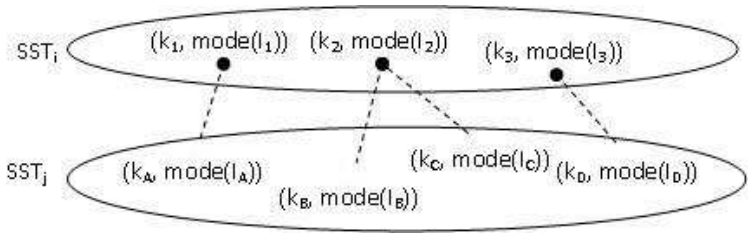


FIG. 4.8 – Cas 1 : SST_i est déductible de SST_j

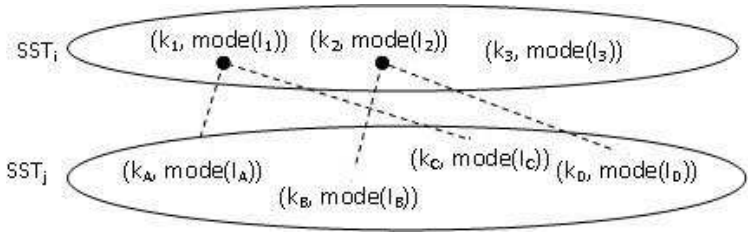


FIG. 4.9 – Cas 2 : SST_i est déductible de SST_j

Nous voyons que SST_i est déductible de SST_j si et seulement si chaque modèle élémentaire impliqué dans SST_j est un des sous-modèles d'un modèle élémentaire impliqués dans SST_i.

Pour le modèle qui se construit au fur et à mesure à différents niveaux d'abstraction, le nombre de SSTs déductibles des autres devient de plus en plus important.

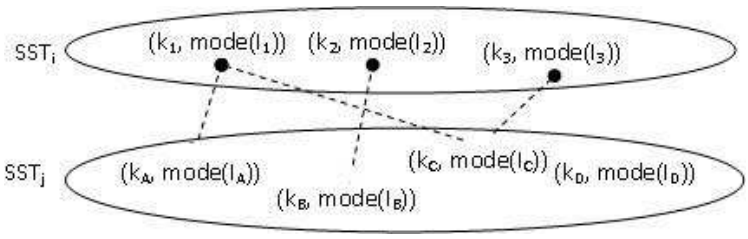


FIG. 4.10 – Cas 3 : SST_i n'est pas déductible de SST_j

Afin de réduire le nombre de calculs de l'automate, il est utile d'enlever des SSTs déductible dans cette étape de génération de tests. Le raisonnement et la détection des SSTs déductibles en terme des contraintes dans l'espace des variables paraissent trop complexe car les contraintes peuvent être de nature différent. De plus, pour calculer les diagnostics, les symptômes collectés sont formulés sous forme de clauses ou de monômes de modes. Pour ces raisons, nous proposons dans la suite une approche qui permette de détecter les SSTs déductibles tel que si un test T_j correspondant à SST_j est inconsistant, les SSTs qui sont déductibles de SST_j ne seront pas considérés dans l'étape de propagation de valeurs. La détection des *SSTs* déductibles sont gérés sous forme d'*un graphe de priorité des SSTs* qui est présenté ensuite.

4.3.3 Graphe de priorité des SSTs

Rappelons que la procédure de décomposition du modèle du système est réalisée en parallèle au niveau des fonctions et au niveau des ressources à l'aide de l'approche structuro-fonctionnelle. Les SSTs sont générés à différents niveaux d'abstraction à partir du modèle comportemental des items. Les fonctions et les ressources sont appelées *items* et ne sont pas distinguées lors de la recherche de SSTs. Ainsi, chaque SST peut être composé par des fonctions et des ressources. Ce sont tous des items dont la description comportementale (contraintes, modes, ...) est disponible.

L'application de la méthode structurelle (Ploix *et al.* [2005b]; Yassine [2008]; Ploix *et al.* [2009, 2010]) au modèle d'abstraction conduit à un ensemble de SSTs, dont certaines sont dites inutiles, ce sont :

1. les SSTs non-féconds. Ces SSTs n'apportent pas d'information à l'étape d'analyse diagnostique et de surcroît, l'existence de ces tests perturbera les résultats de l'analyse diagnostique en altérant les classements. Ces tests sont retirés avant la phase de détection.
2. Les SSTs qui sont déductibles des autres *SSTs* (correspondant aux tests inconsistants). Supposons que SST_i soit déductible de SST_j. Il n'est pas nécessaire de considérer le sous-système testable déductible SST_i à l'étape de test si SST_j le couvrant a été considéré. Cependant, nous ne pouvons pas retirer SST_i avant l'étape de test car il dépend du résultat du test de SST_j. Par exemple, si le test sur SST_j est inconsistant, le test sur SST_i doit être

inconsistant, il n'est pas nécessaire de considérer SST_i. Si le test sur SST_j est consistant, SST_i peut être considéré.

Pour ces raisons, nous proposons de construire le graphe de priorité des SSTs avant l'étape de test. Ce graphe peut être construit automatiquement à partir de l'ensemble des SSTs et du modèle hiérarchie représenté en forme d'un graphe F/R (figure 4.11).

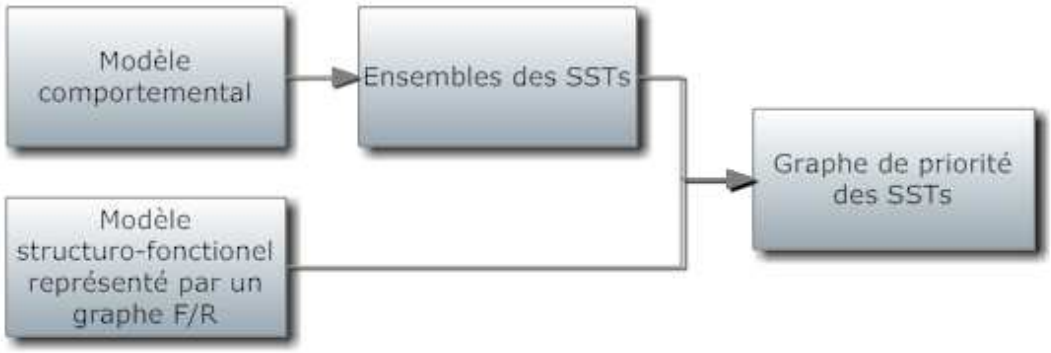


FIG. 4.11 – Construction de graphe de priorité des SSTs)

L'idée est de commencer par enlever les SSTs non-féconds à partir de l'ensemble des SSTs calculés par la méthode structurelle. Ensuite, un arbre permet de gérer l'ordre des tests à effectuer tel que si un test est inconsistant, ses tests déductibles seront ignorés. Un graphe de priorité est illustré par la figure (4.12).

- Chaque noeud est un test qui correspond à un SST.
- Chaque arc est orienté. Un arc qui s'oriente du noeud SST_j vers un noeud SST_i signifie que SST_i (correspondant à T_i) est déductible de SST_j (correspondant à T_j) : on note $SST_j \rightarrow SST_i$. Cela veut dire que s'il y a un défaut dans SST_j, il y a donc un défaut dans SST_i. Dans ce graphe, SST_j est le *prédécesseur* de SST_i et SST_i est le *successeur* de SST_j.

Remarque : Pour $SST_j \rightarrow SST_i$, si T_i est inconsistant, il n'est pas sûr que T_j est inconsistant (le cas de la compensation au niveau des contraintes) mais il existe certainement un défaut dans SST_j. Le défaut dans SST_j peut être détecté si toutes les observations possibles sur SST_j sont considérées.

Construction du graphe de priorité des SSTs

Nous allons montrer dans cette partie comment construire le graphe de priorité des SSTs automatiquement à partir des SSTs générés par la méthode structurelle. Supposons que $\mathbb{S}$ soit un ensemble de SSTs donné par la méthode structurelle au niveau des ressources (ou au niveau des fonctions) après avoir enlevé les $SSTs$ non-féconds. La procédure de construction est la suivante :

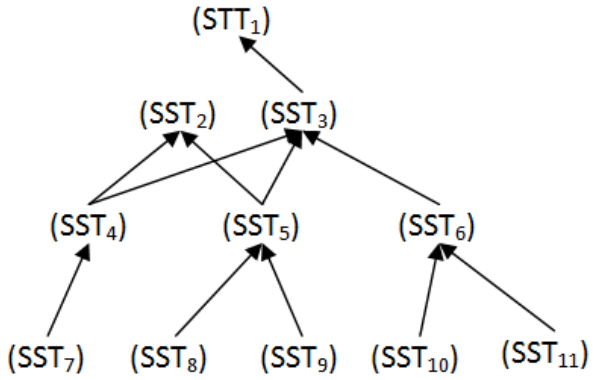


FIG. 4.12 – Graphe de priorité des tests (ou SST)

- Établir les noeuds du graphe
  Chaque noeud est un sous-système testable $SST_j \in \mathbb{S}$. Pour chaque SST_j, on détermine :
  - $Sup(SST_j)$: l'ensemble des modèles élémentaires impliqués dans SST_j et des modèles élémentaires qui sont plus abstraits que ceux impliqués dans SST_j (y compris des modèles plus abstraits donnés par l'intermédiaire de plusieurs niveaux d'abstraction). $Sup(SST_j)$ est construit grâce au graphe d'abstraction sur les modèles élémentaires (figure 2.11, page 43). L'abstraction entre les modèles élémentaires peut être exprimée via l'abstraction sur les items (propriété 1, page 41) ou les relations *ressources d'entrée/fonction* (propriété 3, page 98). La détermination de $Sup(SST_j)$ sera illustrée dans l'exemple 17.
  - $Ddtb(SST_j)$: l'ensemble des SSTs qui sont déductibles de (SST_j) (définition 4.3.4). En utilisant $Sup(SST_j)$, un SST_i est dit *déductible* de SST_j si on a l'ensemble des modèles élémentaires de SST_i est un sous-ensemble de $Sup(SST_j)$, il est écrit par $SST_i \subset Sup(SST_j)$.
- Établir les arcs entre les noeuds
  Une flèche s'oriente d'un noeud SST_j vers un noeud SST_i si
  - SST_i est déductible de SST_j, ou nous pouvons écrire $SST_i \subset Ddtb(SST_j)$.
  - il n'existe pas d'autre noeud SST_k tel que SST_k est déductible de SST_j et SST_k permet de déduire SST_i. On peut écrire, $\nexists SST_k \; tel \; que \; SST_k \subset Ddtb(SST_j) \; et \; SST_i \subset Ddtb(SST_k)$. S'il existe un tel noeud SST_k, le flèche de SST_j vers SST_i sera remplacée par deux flèches : une flèche de SST_j vers SST_k et une autre flèche de SST_k vers SST_i.

Exemple 17. Revenons à l'exemple 14. Après avoir retiré le sous-système testable non-fécond $\{SST_{02}, SST_{03}, SST_{04}, SST_{05}, SST_{07},\}$ (exemple 15), les SSTs restants sont :
$SST_{01} : support = k_{15}[I_{15}], k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_2[I_2], k_{17}[I_{17}]$
$SST_{06} : support = k_9[I_9], k_{14}[I_{14}], k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2]$
$SST_{08} : support = k_{15}[I_{15}], k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2], k_{17}[I_{17}]$
$SST_{09} : support = k_8[I_8], k_5[I_5], k_3[I_3], k_2[I_2], k_1[I_1]$
$SST_{10} : support = k_8[I_8], k_5[I_5], k_4[I_4], k_3[I_3], k_2[I_2]$

Les relations hiérarchiques entre les items sont présentées sur la figure (4.13).

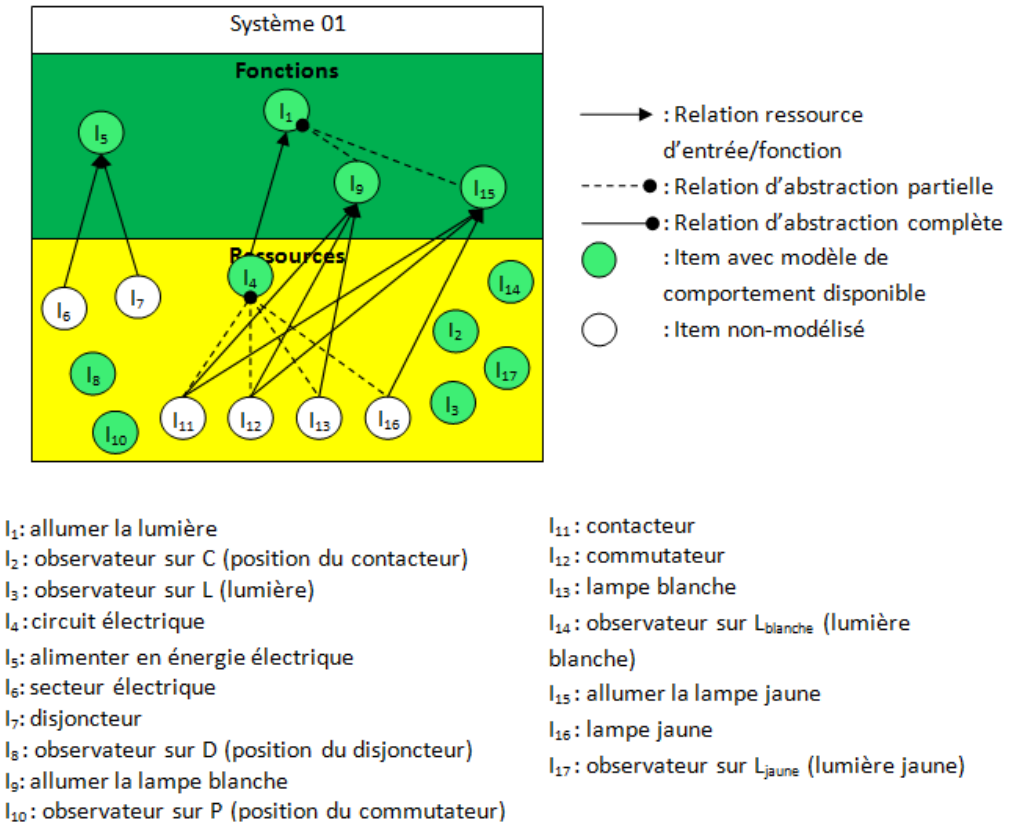


FIG. 4.13 – Modélisation FIS

A partir de l'ensemble $\{SST_{01}, SST_{06}, SST_{08}, SST_{09}, SST_{10}\}$ et du graphe structuro-fonctionnel (figure 4.13), nous construisons le graphe de priorité des $SST s$. En se basant sur le graphe F/R représenté sur la figure 4.13, un graphe qui représente les relations d'abstraction entre les modèles élémentaires impliqués dans le système est donné sur la figure 4.14 grâce aux propriétés 1 (page 41) et 3 (page 98).

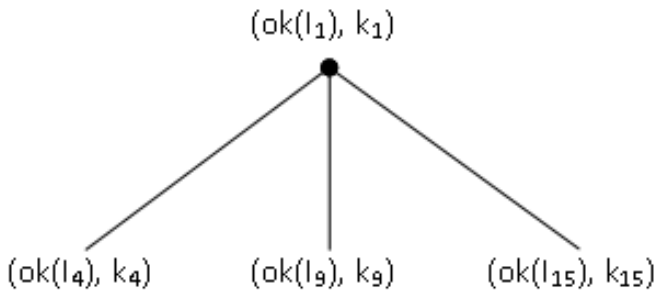


FIG. 4.14 – Modélisation FIS

En s'appuyant sur l'abstraction sur les modèles élémentaires présentée sur la figure 4.14, nous déterminons $Sup(SST_i)$ pour chaque SST_i. Pour le modèle de fonctionnement normal, un

modèle élémentaire $(mode(I_1), k_1)$ est noté simplement par $k_1[I_1]$ dans DXLab. Nous avons :
$Sup(SST_{01}) = \{k_{15}[I_{15}], k9[I9], k_{14}[I_{14}], k_{10}[I_{10}], k_2[I_2], k_{17}[I_{17}], k_1[I_1], \}$
$Sup(SST_{06}) = \{k_9[I_9], k_{14}[I_{14}], k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2], k_1[I_1]\}$
$Sup(SST_{08}) = \{k_{15}[I_{15}], k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2], k_{17}[I_{17}], k_1[I_1]\}$
$Sup(SST_{09}) = \{k_8[I_8], k_5[I_5], k_3[I_3], k_2[I_2], k_1[I_1]\}$
$Sup(SST_{10}) = \{k_8[I_8], k_5[I_5], k_4[I_4], k_3[I_3], k_2[I_2], k_1[I_1]\}$

Nous déterminons ensuite, l'ensemble déductible $Ddtb(SST_i)$ de chaque SST_i : $Ddtb(SST_{01}) = \{\emptyset\}$, $Ddtb(SST_{06}) = \{\emptyset\}$, $Ddtb(SST_{08}) = \{\emptyset\}$, $Ddtb(SST_{09}) = \{\emptyset\}$, $Ddtb(SST_{10}) = \{SST_9\}$.

Nous voyons que $SST_{09} \in Ddtb(SST_{10})$. Un graphe de priorité des SSTs est donc donné sur la figure (4.15).

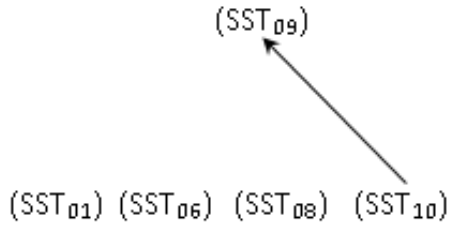


FIG. 4.15 – Graphe de priorité des tests

Utilisation du graphe de priorité des SSTs

L'étape de test est réalisée par un simulateur de la façon suivante :

- Traitement des noeuds : si un SST_i est choisi pour être effectué, il sera étiqueté par P si le test est consistant et par N si le test est inconsistant. Si SST_i est inconsistant, tous ses successeurs seront retirés.
  En effet, les successeurs de SST_i sont, dans ce cas, des tests déductibles de SST_i. Selon la définition 4.3.4 (page 113), les successeurs de SST_i seront certainement des tests inconsistants. De plus, il sont non-minimaux en comparaison avec SST_i. Les tests non-minimaux ne seront pas considérés dans l'analyse diagnostique. L'enlèvement des successeurs de SST_i permet de réduire le temps de calcul.
  Revenons au graphe de priorité présenté sur la figure (4.12). Si SST_9 est inconsistant, il est étiqueté par N. Les successeurs SST_5, SST_2, SST_3, SST_1 seront donc retirés avec tous leurs successeurs.
- Choix d'un test à effectuer : un test associé à un noeud sera choisi pour la prochaine simulation si :
  - il n'a pas été retiré.

- il n'a pas de prédécesseurs ou tous ses prédécesseurs sont étiquetés. Les noeuds étiquetés (P/N) sont des noeuds ayant été affectés.

  Sur l'exemple de la figure (4.12), si SST_9 est déjà étiqueté, le prochain test sera choisi parmi SST_7, SST_8, SST_{10} ou SST_{11}.
- Arrêt des tests : l'étape de test s'arrête quand tous les noeuds sont étiquetés ou retirés.

Conclusion

Dans ce problème de diagnostic avec interaction dans la phase de modélisation, le modèle du système est composé au fur et à mesure. En se basant sur le modèle structuro-fonctionnel formulé dans les sections 2.3 et 2.4, les modèles élémentaires (une contrainte qui décrit un mode de l'item) sont ajoutés pour décrire le modèle comportemental des items. Les SSTs qui s'appuient sur les modèles élémentaires sont générés automatiquement par l'automate pour détecter et collecter les symptômes. Les diagnostics sont alors calculés. Les relations hiérarchiques entre les items décrits aux différents niveaux d'abstraction peuvent conduire à des SSTs inutiles qui ne permettent pas de collecter de nouveaux symptômes et qui vont perturber les résultats de diagnostic. En se basant sur la formulation de *l'abstraction sur les modèles élémentaires* qui se rapporte à l'abstraction sur les items et aux relations *ressources d'entrée / fonction*, les *SSTs non-féconds* et les *SSTs* qui sont *déductibles* des autres *SSTs* sont enlevés afin de ne pas perturber les résultats de diagnostic. Pour gérer les SSTs, un graphe de priorité des SSTs a été conçu.

4.4 Application

Revenons à l'exemple du système d'éclairage dans le bâtiment. Nous allons détailler cet exemple avec plusieurs interactions pour illustrer comment un processus de diagnostic avec interactions durant la phase de modélisation se déroule. Pour distinguer ce problème du problème précédent avec interaction durant la phase de test, cet exemple suppose que l'expert ne peut pas déterminer si un test est vrai ou faux. Il a besoin de l'outil d'aide au diagnostic pour générer des symptômes.

Dans ce problème de diagnostic interactif dans la phase de modélisation, l'outil d'aide au diagnostic peut générer automatiquement des symptômes en s'appuyant sur le modèle structuro-fonctionnel et sur le modèle de comportement d'un système qui est construit itérativement à différents niveaux d'abstraction. Ceci est la contribution principale de cette partie. Une fois que les symptômes sont collectés, les diagnostics peuvent être calculés par l'approche que nous avons présentée dans le chapitre 3. Dans cet exemple, nous allons nous concentrer sur l'étape de détection des symptômes à chaque interaction. Pour simplifier, les défaut des capteurs sont ignorés dans cet exemple.

Tout d'abord, un client signale que le système d'éclairage dans son bâtiment est défectueux. L'expert commence le processus de diagnostic pour localiser les défauts.

1. **Interaction 1 : Construction du modèle par l'expert**

   L'expert ajoute la fonction I_1 (*allumer la lumière*). Le modèle de comportement de I_1 est décrit par le modèle élémentaire $(ok(I_1), k_1)$. k_1 est représenté par le tableau 4.6. Il ajoute des observateurs I_2 et I_3 qui sont décrits respectivement par les modèles élémentaires $(ok(I_2), k_2 : C = \widetilde{C})$ et $(ok(I_3), k_3 : L = \widetilde{L})$. C est la position d'un contacteur, *dom(C)={on, off}*. L est la lumière avec *dom(L)={présent, absent}*.

   V est la tension d'alimentation impliquée dans la fonction I_5 (*alimenter en énergie électrique*) avec *domV={présent, absent}*. I_5 est décrit par le modèle élémentaire $(ok(I_5), k_5)$ où k_5 est représenté par le tableau 4.7. La variable D est la position du disjoncteur avec *dom(D)={fermé, ouvert}*. I_6 (*secteur électrique*) et I_7 (*disjoncteur* sont les deux ressources d'entrée de la fonction I_5. I_8 est un observateur sur la position du disjoncteur. I_8 est décrit par $(ok(I_8), k_8 : D = \widetilde{D})$.

Modes	V (tension)	C (position du contacteur)	L (lumière)
ok	présent	on	présent

TAB. 4.6 – Contrainte k_1 (fonction allumer la lumière)

Modes	D (position du contacteur)	V V (tension)
ok	fermé	présent
ok	ouvert	absent

TAB. 4.7 – Contrainte k_5 (fonction alimenter en énergie électrique)

   Le modèle FIS du système à cette étape est présenté sur la figure (4.16). Les connections entre les variables d'une même grandeur physique sont présentées sur la figure (4.17). Notons que nous considérons dans cet exemple le cas simple où les connections entre les variables liées à une même grandeur sont complètes.

   Supposons que l'expert alimente l'automate (le système d'aide au diagnosic) avec les observations suivantes $\widetilde{D}$=*fermé*, $\widetilde{C} = on$ et $\widetilde{L} = absent$.

2. **Interaction 1 : Génération automatique des symptômes**

   Le seul sous-système testable minimal qui peut être calculé est donné par :
   SST_{01} : $support = k_8[I_8], k_5[I_5], k_3[I_3], k_2[I_2], k_1[I_1]$
   $Formula = ((k_5 - D - k_8) - V - ((k_1 - L - k_3) - C - k_2))$

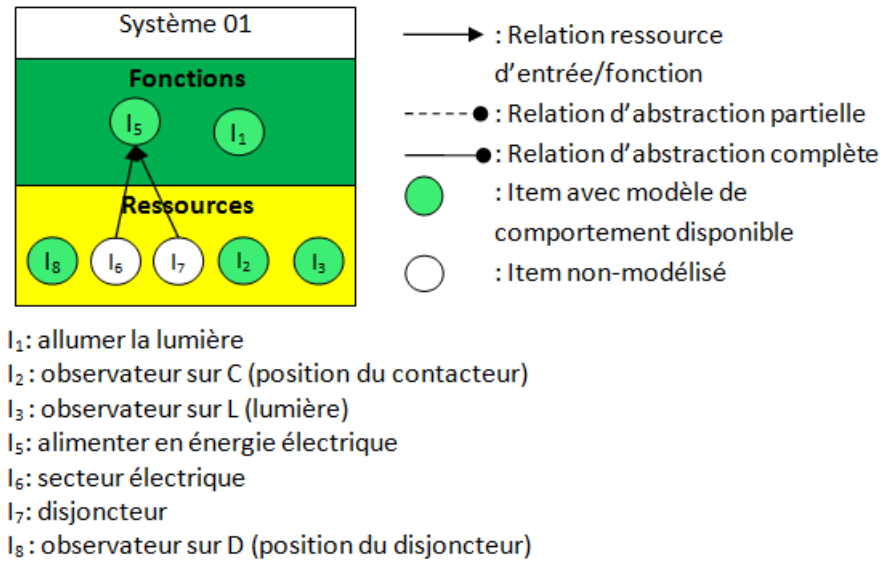


FIG. 4.16 – Modèle FIS du système à l'interaction 1

Aucune SST non-fécond n'est détecté.

Le symptôme (SST_i correspond à un test T_i) est généré par la méthode forward checking en se basant sur le problème de satisfaction de contraintes (voir l'annexe C) : $\top(T_1)$.

3. **Interaction 1 : Calcul de diagnostics**

   Pour calculer des diagnostics à partir des symptômes, l'approche présentée dans le chapitre 3 a été utilisée. L'automate complète des abstractions partielles si elles existent en ajoutant des items virtuels. Puis il transforme le modèle structuro-fonctionel donné par la figure 4.16 en un ensemble de propositions logiques.

$$cfm(I_5) \leftrightarrow cfm(I_6) \vee cfm(I_7) \tag{4.4.1}$$

   Puis la proposition 4.4.1 est remplacée dans l'explication du symptôme $\top(T_1)$ pour calculer les diagnostics. La table de signature du test est donnée par le tableau 4.8.

	$ok\ (I_1)$	$ok\ (I_6)$	$ok\ (I_7)$
T_1	1	1	1

TAB. 4.8 – Table de signature donnée à interaction 1

   Les diagnostics sont donnés par : $\{cfm\ (I_1)\}$, $\{cfm\ (I_6)\}$, $\{cfm\ (I_7)\}$.

   Les mesures de coïncidence des défauts sont données par le tableau 4.9.

4. **Interaction 2 : Construction du modèle par l'expert**

   En s'appuyant sur les mesures de coïncidence des défauts, l'expert continue à décomposer le système. I_4 (*circuit électrique*) est une ressource d'entrée de la fonction I_1. Le

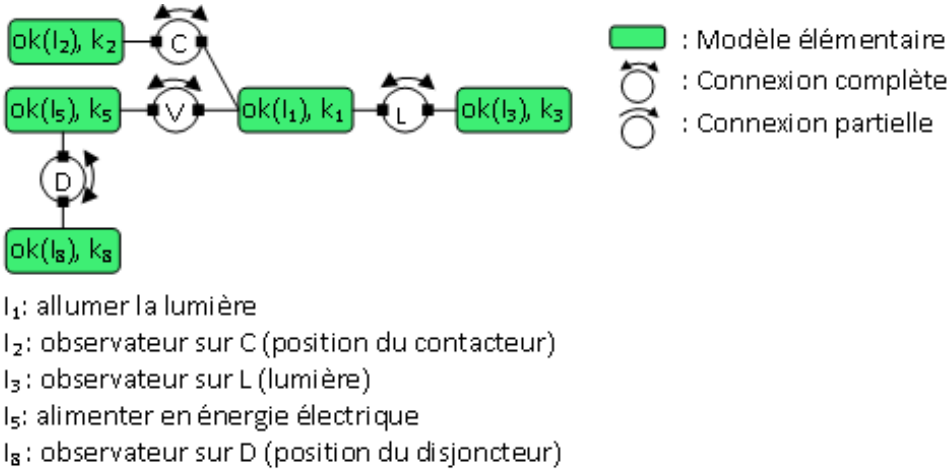


FIG. 4.17 – Modèle de comportement du système à interaction 1

Signature théorique	Signature effective	Mesure de coïncidence
$\sigma_T(cfm\ (I_1)) = (1)$	$\sigma_T^* = (1)$	$\mu_T^c(cfm\ (I_1)) = 0.00$
$\sigma_T(cfm\ (I_6)) = (1)$	$\sigma_T^* = (1)$	$\mu_T^c(cfm\ (I_1)) = 0.00$
$\sigma_T(cfm\ (I_7)) = (1)$	$\sigma_T^* = (1)$	$\mu_T^c(cfm\ (I_1)) = 0.00$

TAB. 4.9 – Mesures de coïncidence des défauts à l'interaction 1

modèle de comportement de I_4 est décrit par le modèle élémentaire $(ok(I_4), k_4)$ avec k_4 représenté par le tableau 4.10.

Modes	V (tension)	C (position du contacteur)	L (lumière)
ok	présent	off	absent
ok	présent	on	présent
ok	absent	off	absent
ok	absent	on	absent

TAB. 4.10 – Contrainte k_4 (circuit électrique

L'expert découvre le système. Le bâtiment peut être éclairé en lumière blanche ou en lumière jaune. L'expert enrichit le modèle en décomposant la fonction I_1 (*allumer la lumière*) en sous-fonction I_9 (*allumer la lumière blanche*) pour tester. I_9 est décrit par le modèle élémentaire $(ok(I_9), k_9)$. k_9 est représenté par le tableau 4.11. P est la position du commutateur avec *dom(P)={*x_{blanc}, x_{jaune}*}*. I_{11} (*contacteur*), I_{12} (*commutateur*) et I_{13} (*lampe blanche*) sont les ressources d'entrée de la fonction I_9. I_{10} est un observateur sur la position du commutateur. I_{10} est décrit par $(ok(I_{10}), k_{10} :\ P = \widetilde{P})$. I_{14} est un observateur sur la position du commutateur. I_{14} est décrit par $(ok(I_{14}), k_{14} :\ L_{blanc} = \widetilde{L_{blanc}})$.

Modes	V (tension)	C (position du contacteur)	P (position du commutateur)	L_{blanc} (lumière blache)
ok	*prsent*	on	x_{blanc}	*prsent*

TAB. 4.11 – Contrainte k_9 (fonction allumer la lumière blanche)

L'expert vérifie la position du commutateur. Les observations mises à jour à cette étape sont : $\widetilde{D}$=*fermé*, $\widetilde{C}$= *on*, $\widetilde{L}$= *absent*, $\widetilde{P}$=x_{blanc}, $\widetilde{L_{blanc}} = absent$.

Le modèle FIS du système à cette étape est représenté sur la figure 4.18.

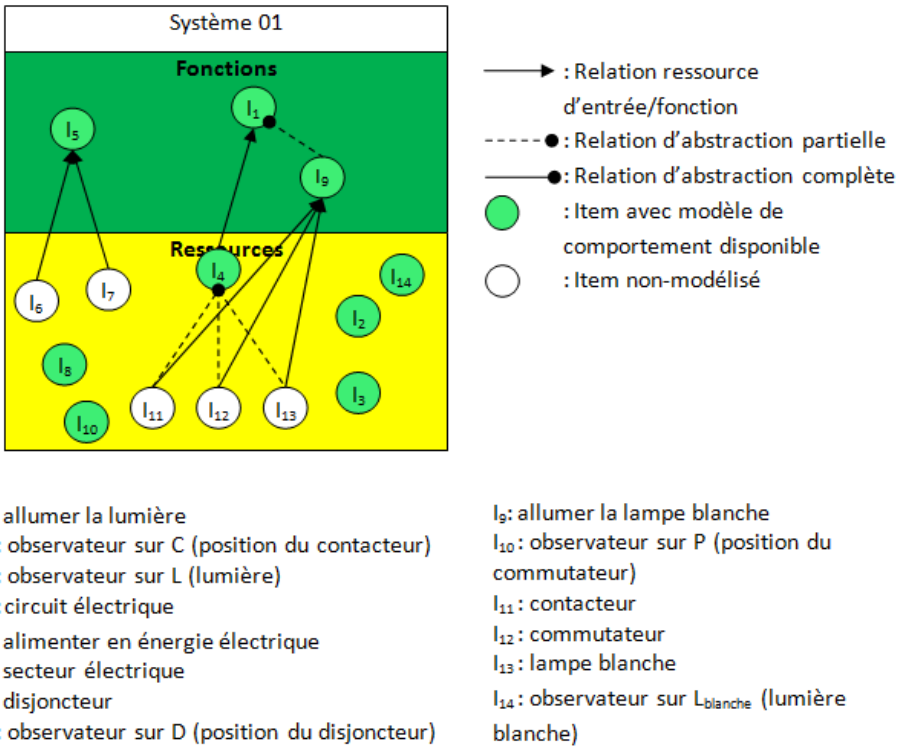


FIG. 4.18 – Modèle FIS du système à l'interaction 2

Les connexions entre les variables d'une même grandeur physique sont présentées sur la figure 4.19.

5. **Interaction 2 : Génération automatique des symptômes**

Les sous-systèmes testables minimaux sont

SST_{01} : $support = k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_4[I_4], k_3[I_3], k_2[I_2]$

$Formula = (((k_4 - L - k_3) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$

SST_{02} : $support = k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_3[I_3], k_2[I_2], k_1[I_1]$

$Formula = (((k_1 - L - k_3) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$

SST_{03} : $support = k_4[I_4], k_3[I_3], k_2[I_2], k_1[I_1]$

$Formula = (((k_4 - L - k_3) - C - k_2) - V - ((k_1 - L - k_3) - C - k_2))$

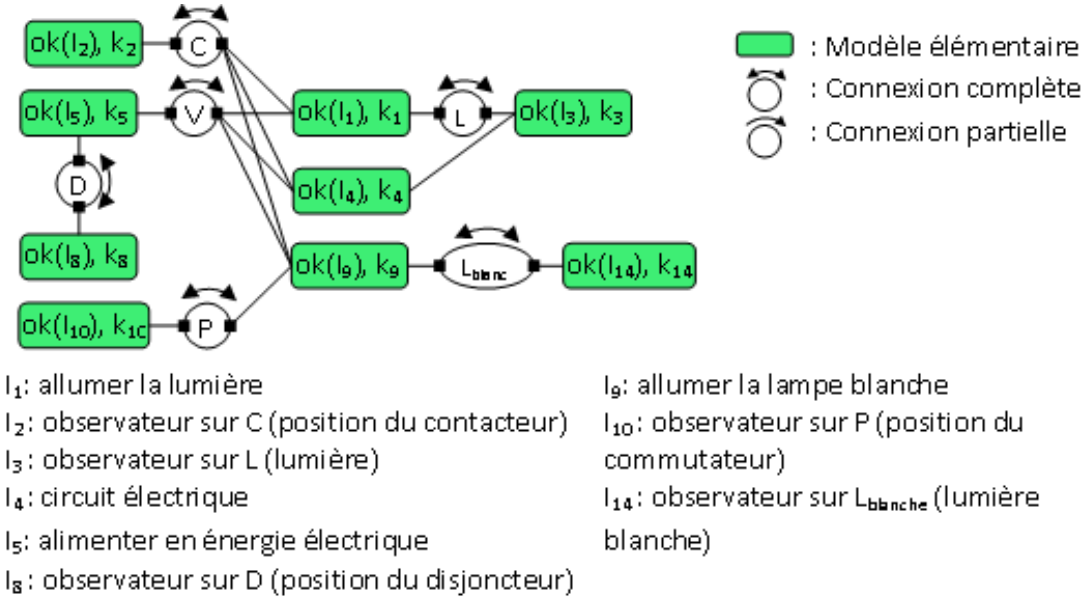


FIG. 4.19 – Modèle de comportement du système à l'interaction 2

SST_{04} : $support = k_9[I_9], k_{14}[I_{14}], k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2]$
$Formula = ((k_5 - D - k_8) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$
SST_{05} : $support = k_8[I_8], k_5[I_5], k_3[I_3], k_2[I_2], k_1[I_1]$
$Formula = ((k_5 - D - k_8) - V - ((k_1 - L - k_3) - C - k_2))$
SST_{06} : $support = k_8[I_8], k_5[I_5], k_4[I_4], k_3[I_3], k_2[I_2]$
$Formula = ((k_5 - D - k_8) - V - ((k_4 - L - k_3) - C - k_2))$

Dans cet ensemble, les SST non-féconds sont : SST_{01} (selon règle 3, page 110), SST_{02} (selon règle 1, page 110) et SST_{03} (selon règle 2, page 111).
Après avoir retiré les SSTs non-féconds, le graphe de priorité des SSTs est donné par la figure 4.20.

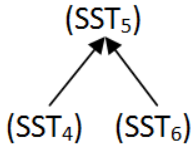


FIG. 4.20 – Graphe de priorité des tests à l'interaction 2

Les symptômes générés sont : $\top(T_4)$, $\top(T_6)$. SST_{05} n'est pas considéré car il peut être déductible de $\top(T_4)$ ou de $\top(T_6)$.

6. **Interaction 2 : Calcul des diagnostics**
   L'automate complète des abstractions partielles en ajoutant des items virtuels : $IV_{2.1} = I_4 \setminus \{I_{11}, I_{12}, I_{13}\}$.

Puis il transforme le modèle structuro-fonctionel donné par la figure 4.18 en un ensemble de propositions logiques 4.4.2.

$$\begin{aligned} cfm(I_5) &\leftrightarrow cfm(I_6) \vee cfm(I_7) \\ cfm(I_1) &\leftrightarrow cfm(I_4) \\ cfm(I_9) &\leftrightarrow cfm(I_{11}) \vee cfm(I_{12}) \vee cfm(I_{13}) \\ cfm(I_4) &\leftrightarrow cfm(I_{11}) \vee cfm(I_{12}) \vee cfm(I_{13}) \vee cfm(IV_{2.1}) \end{aligned} \tag{4.4.2}$$

Les modes parents impliqués dans l'explication des symptômes sont remplacés grâce à l'ensemble de propositions 4.4.2 (les défauts des capteurs sont ignorés dans cet exemple), nous avons :

$$\begin{aligned} Expl(\top(T_4)): &\ cfm\ (R_{06}) \vee cfm\ (R_{07}) \vee cfm\ (R_{11}) \vee cfm\ (R_{12}) \vee cfm\ (R_{13}) \\ Expl(\top(T_6)): &\ cfm\ (R_{06}) \vee cfm\ (R_{07}) \vee cfm\ (R_{11}) \vee cfm\ (R_{12}) \vee cfm\ (R_{13}) \end{aligned} \tag{4.4.3}$$

La table de signature des tests à cette itération est donné par le tableau 4.12.

	$ok\ (R_6)$	$ok\ (R_7)$	$ok\ (R_{11})$	$ok\ (R_{12})$	$ok\ (R_{13})$	$ok\ (IV_{2.1})$
T_1	1	1	1	1	1	0
T_2	1	1	1	1	1	1

TAB. 4.12 – Table de signature donnée à l'interaction 2

Les diagnostics sont donnés par :
$\{cfm(I_6)\}$, $\{cfm(I_7)\}$, $\{cfm(I_{11})\}$, $\{cfm(I_{12})\}$, $\{cfm(I_{13})\}$
Les mesures de coïncidence des défauts sont données par le tableau 4.13.

Signature théorique	Signature effective	Mesure de coïncidence
$\sigma_T(cfm\ (I_6)) = (1\ \ 1)$	$\sigma_T^* = (1\ \ 1)$	$\mu_T^c(cfm\ (I_6)) = 0.00$
$\sigma_T(cfm\ (I_7)) = (1\ \ 1)$		$\mu_T^c(cfm\ (I_7)) = 0.00$
$\sigma_T(cfm\ (I_{11})) = (1\ \ 1)$		$\mu_T^c(cfm\ (I_{11})) = 0.00$
$\sigma_T(cfm\ (I_{12})) = (1\ \ 1)$		$\mu_T^c(cfm\ (I_{12})) = 0.00$
$\sigma_T(cfm\ (I_{13})) = (1\ \ 1)$		$\mu_T^c(cfm\ (I_{13})) = 0.00$

TAB. 4.13 – Mesures de coïncidence des défauts à l'interaction 2

7. **Interaction 3 : Construction du modèle par l'expert**

L'expert décompose et enrichit le modèle en décomposant la fonction I_1 *: allumer la lumière* en autre sous-fonction I_{15} (*allumer la lumière jaune*) pour tester. I_{15} est décrit par le modèle élémentaire $(ok(I_{14}), k_{15})$. k_{15} est représenté sur le tableau (4.14). I_{11}

Modes	V	C	P	L_{jaune}
ok	*présent*	on	x_2	*présent*

TAB. 4.14 – Contrainte k_{15} (fonction allumer la lumière jaune)

(*contacteur*), I_{12} (*commutateur*) et I_{16} (*lampe jaune*) sont les ressources d'entrée de la fonction I_{15}. I_{17} est décrit par $(ok(I_{17}), k_{17} : \widetilde{L_{jaune} = L_{jaune}})$.

L'expert change la position du commutateur. Les observations mises à jour à cette étape sont : $\widetilde{D}$=*fermé*, $\widetilde{C} = on$, $\widetilde{L} = absent$, $\widetilde{P} = x_{jaune}$, $\widetilde{L_{blanc}} = absent$, $\widetilde{L_{jaune}} = absent$.

Le modèle FIS du système à cette étape est représenté sur la figure 4.21.

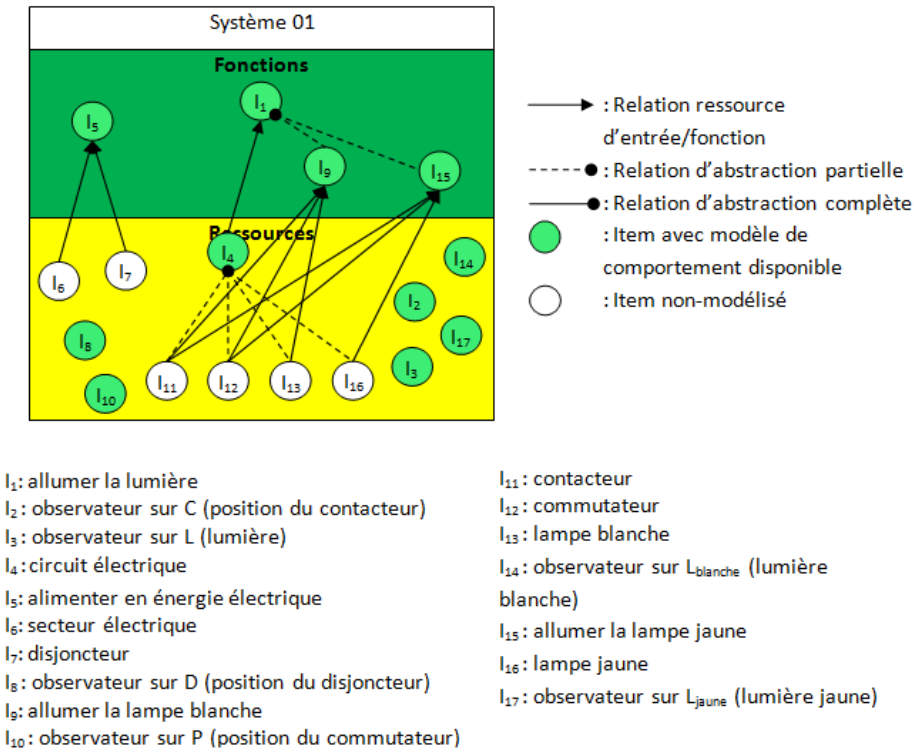


FIG. 4.21 – Modèle FIS du système à interaction 3

Les connexions entre les variables d'un même grandeur physique sont présentées sur la figure 4.22.

8. **Interaction 3 : Génération automatique des symptômes**

Les sous-systèmes testables minimaux sont :

$SST_{01} : support = k_{15}[I_{15}], k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_2[I_2], k_{17}[I_{17}]$

$Formula = ((((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$

$SST_{02} : support = k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_3[I_3], k_2[I_2], k_1[I_1]$

$Formula = (((k_1 - L - k_3) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$

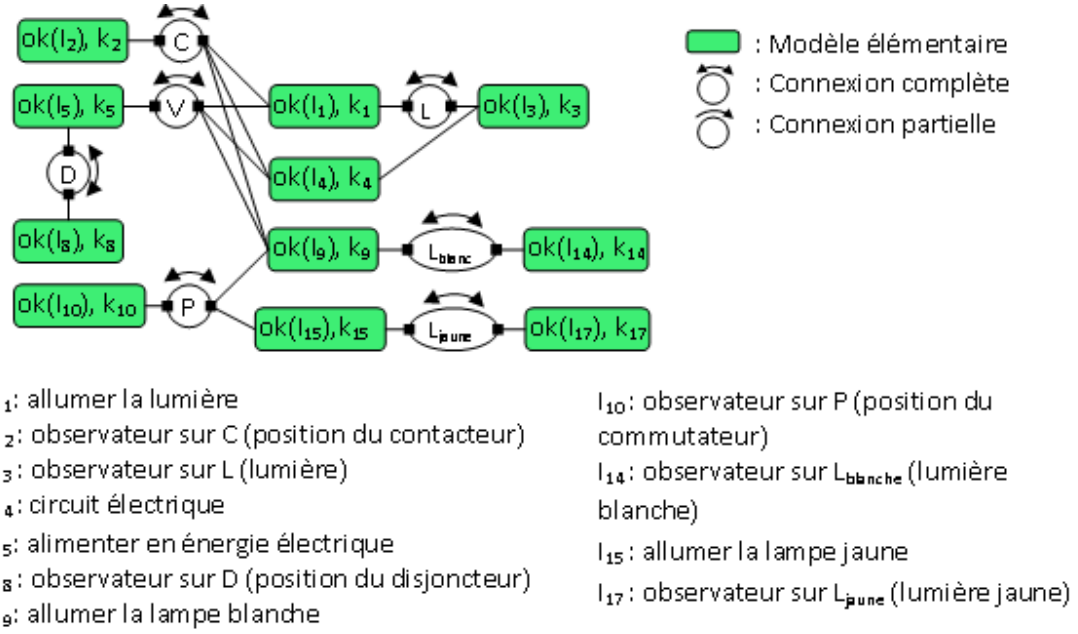


FIG. 4.22 – Modèle de comportement du système à interaction 3

$SST_{03} : support = k_{15}[I_{15}], k_{10}[I_{10}], k_3[I_3], k_2[I_2], k_1[I_1], k_{17}[I_{17}]$
$Formula = (((k_1 - L - k_3) - C - k_2) - V - (((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2))$
$SST_{04} : support = k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_4[I_4], k_3[I_3], k_2[I_2]$
$Formula = (((k_4 - L - k_3) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$
$SST_{05} : support = k_{15}[I_{15}], k_{10}[I_{10}], k_4[I_4], k_3[I_3], k_2[I_2], k_{17}[I_{17}]$
$Formula = (((k_4 - L - k_3) - C - k_2) - V - (((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2))$
$SST_{06} : support = k_9[I_9], k_{14}[I_{14]}, k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2]$
$Formula = ((k_5 - D - k_8) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$
$SST_{07} : support = k_4[I_4], k_3[I_3], k_2[I_2], k_1[I_1]$
$Formula = (((k_4 - L - k_3) - C - k_2) - V - ((k_1 - L - k_3) - C - k_2))$
$SST_{08} : support = k_{15}[I_{15}], k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2], k_{17}[I_{17}]$
$Formula = ((k_5 - D - k_8) - V - (((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2))$
$SST_{09} : support = k_8[I_8], k_5[I_5], k_3[I_3], k_2[I_2], k_1[I_1]$
$Formula = ((k_5 - D - k_8) - V - ((k_1 - L - k_3) - C - k_2))$
$SST_{10} : support = k_8[I_8], k_5[I_5], k_4[I_4], k_3[I_3], k_2[I_2]$
$Formula = ((k_5 - D - k_8) - V - ((k_4 - L - k_3) - C - k_2))$

Dans cet ensemble, les SST non-féconds sont : SST_{02} (selon règle 1, page 110), SST_{03} (selon règle 1), SST_{04} (selon règle 3), SST_{05} (selon règle 3), SST_{07} (selon règle 2).

Après avoir retiré les SSTs non-féconds, le graphe de priorité des SSTs est donné sur la figure 4.23.

Les symptômes sont : $\top(T_1)$, $\top(T_6)$, $\top(T_8)$, $\top(T_{10})$. SST_{09} n'est pas considéré car il

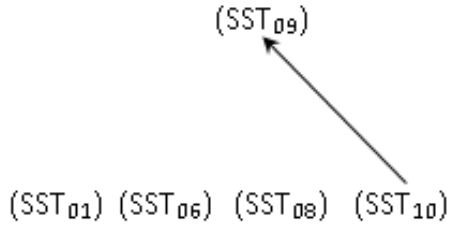


FIG. 4.23 – Graphe de priorité des tests à interaction 3

peut être déductible de $\top(T_6)$, $\top(T_8)$ ou de $\top(T_{10})$.

9. **Interaction 3 : Calcul de diagnostics**
   L'automate complète les abstractions partielles en ajoutant des items virtuels : $IV_{3.1} = I_4 \setminus \{I_{11}, I_{12}, I_{13}, I_{16}\}$. Notons que $IV_{3.1} = IV_{2.1} \setminus \{I_{16}\}$.
   Puis il transforme le modèle structuro-fonctionel donné par la figure 4.18 en un ensemble de propositions logiques 4.4.4.

$$\begin{aligned} cfm(I_5) &\leftrightarrow cfm(I_6) \vee cfm(I_7) \\ cfm(I_1) &\leftrightarrow cfm(I_4) \\ cfm(I_9) &\leftrightarrow cfm(I_{11}) \vee cfm(I_{12}) \vee cfm(I_{13}) \\ cfm(I_9) &\leftrightarrow cfm(I_{11}) \vee cfm(I_{12}) \vee cfm(I_{16}) \\ cfm(I_4) &\leftrightarrow cfm(I_{11}) \vee cfm(I_{12}) \vee cfm(I_{13}) \vee cfm(I_{16}) \vee cfm(IV_{3.1}) \end{aligned} \tag{4.4.4}$$

   Les modes parents impliqués dans l'explication des symptômes sont remplacés grâce à l'ensemble de propositions 4.4.4 (les défauts des capteurs sont ignorés dans cet exemple), nous avons :

$$\begin{aligned} Expl(\top(T_{01})) :&\ \ cfm\,(R_{11}) \vee cfm\,(R_{12}) \vee cfm\,(R_{13}) \vee cfm\,(R_{16}) \\ Expl(\top(T_{06})) :&\ \ cfm\,(R_{06}) \vee cfm\,(R_{07}) \vee cfm\,(R_{11}) \vee cfm\,(R_{12}) \vee cfm\,(R_{13}) \\ Expl(\top(T_{08})) :&\ \ cfm\,(R_{06}) \vee cfm\,(R_{07}) \vee cfm\,(R_{11}) \vee cfm\,(R_{12}) \vee cfm\,(R_{13}) \ldots \\ &\qquad \cdots \vee cfm\,(R_{16}) \\ Expl(\top(T_{10})) :&\ \ cfm\,(R_{06}) \vee cfm\,(R_{07}) \vee cfm\,(R_{11}) \vee cfm\,(R_{12}) \vee cfm\,(R_{13}) \ldots \\ &\qquad \cdots \vee cfm\,(R_{16}) \vee cfm\,(IV_{3.1}) \end{aligned} \tag{4.4.5}$$

   Après le remplacement, la table de signature des tests est donnée par le tableau 4.15.
   Les diagnostics sont donnés par :
   $\{cfm(I_{11}\}, \{cfm(I_{12}\}$
   $\{cfm\,(I_{13}) \wedge cfm\,(I_6)\}$, $\{cfm\,(I_{13}) \wedge cfm\,(I_7)\}$, $\{cfm\,(I_{16}) \wedge cfm\,(I_6)\}$, $\{cfm\,(I_{16}) \wedge cfm\,(I_7)\}$

	$ok\ (R_6)$	$ok\ (R_7)$	$ok\ (R_{11})$	$ok\ (R_{12})$	$ok\ (R_{13})$	$ok\ (R_{16})$	$ok\ (IV_{3.1})$
T_1	0	0	1	1	1	1	0
T_6	1	1	1	1	1	0	0
T_8	1	1	1	1	0	1	0
T_{10}	1	1	1	1	1	1	1

TAB. 4.15 – Table de signature donnée à interaction 3

Signature théorique	Signature effective	Mesure de coïncidence
$\sigma_T(cfm\ (I_{11}) = (1\ \ 1\ \ 1\ \ 1)$	$\sigma^*_T = (1\ \ 1\ \ 1\ \ 1)$	$\mu^c_T(cfm\ (I_{11})) = 0.00$
$\sigma_T(cfm\ (I_{12}) = (1\ \ 1\ \ 1\ \ 1)$		$\mu^c_T(cfm\ (I_{12})) = 0.00$
$\sigma_T(cfm\ (I_{13}) \wedge cfm\ (I_6)) = (1\ \ 1\ \ 1\ \ 1)$		$\mu^c_T(cfm\ (I_{13}) \wedge cfm\ (I_6)) = 0.00$
$\sigma_T(cfm\ (I_{13}) \wedge cfm\ (I_7)) = (1\ \ 1\ \ 1\ \ 1)$		$\mu^c_T(cfm\ (I_{13}) \wedge cfm\ (I_7)) = 0.00$
$\sigma_T(cfm\ (I_{16}) \wedge cfm\ (I_6)) = (1\ \ 1\ \ 1\ \ 1)$		$\mu^c_T(cfm\ (I_{16}) \wedge cfm\ (I_6)) = 0.00$
$\sigma_T(cfm\ (I_{16}) \wedge cfm\ (I_7)) = (1\ \ 1\ \ 1\ \ 1)$		$\mu^c_T(cfm\ (I_{16}) \wedge cfm\ (I_7)) = 0.00$

TAB. 4.16 – Mesures de coïncidence des défauts à l'interaction 3

Les mesures de coïncidence des défauts sont données par le tableau 4.16.

De la même manière, l'expert continue la procédure de diagnostic jusqu'à ce qu'un ou plusieurs défauts aient été localisés.

4.5 Conclusion

Dans ce chapitre, nous avons présenté un processus de diagnostic avec interactions homme-automate en phase de modélisation. Nous nous appuyons sur une modélisation structuro-fonctionnelle doublée d'une modélisation comportementale pour décrire le système au fur et à mesure à différents niveaux de détail. Dans ce modèle, certains liens *fonction/ressource* (les liens 2b et 3b sur la figure 4.3) sont ignorés. Du point de vue du modèle de comportement, ces liens peuvent être exprimés par les connexions entre les variables impliqués dans les modèles élémentaires. Au niveau de l'algorithme de calcul, la contribution de cette partie se focalise sur la tâche de génération automatique des symptômes pour un modèle qui est décrit à différents niveaux d'abstraction. Comme la propagation des valeurs pose des difficultés liées à la coexistence de différents niveaux d'abstraction, nous avons divisé la tâche de génération des symptômes en deux sous-tâches : génération des SST par la méthode structurelle et propagation de valeurs au sein de chaque SST pour générer des symptômes. Nous avons appliqué tout d'abord la méthode structurelle pour chercher des sous-systèmes testables en gérant les relations hiérarchiques entre les items pour qu'elles ne perturbent pas les résultats de diagnostics. Pour ce but, nous avons introduit la notion de *SST non-fécond* et *le graphe de priorité des SSTs*. Après avoir retiré des SSTs non-féconds, la gestion des SSTs par le graphe de priorité

permet d'éviter au mieux l'altération du classement des diagnostics. Ensuite, la méthode de propagation de valeur inspirée du problème de satisfaction de contrainte (CSP) est appliqué ensuite pour chaque SST pour collecter des symptômes (annexe C). Une fois que les symptômes sont générés, les diagnostics sont calculés par l'approche que nous avons présentée dans le chapitre 3.

Chapitre 5

Diagnostic itératif basé sur la connaissance experte implicite durant la phase d'analyse diagnostique

5.1 Introduction

Dans cette partie, nous nous focalisons sur le problème de diagnostic avec interactions homme-automate durant la phase de diagnostic pour exploiter l'expertise implicite. Comme nous l'avons présenté dans l'introduction générale de ce manuscrit, les chapitres 3 et 4 résolvent les difficultés liés à la complexité du modèle tandis que ce chapitre vise à résoudre les difficultés liées à l'expertise non-formulée. En comparaison avec les deux autres types d'interaction étudiés, ce cas n'induit pas de changements sur la conception du modèle du système. Le modèle du système est établi avant que le processus de diagnostic ne commence. L'expertise non-formulée est exploitée au fur et à mesure dans la phase de diagnostic pour affiner les résultats de diagnostic.

Certaines connaissances expertes sont formalisables à travers le modèle et les tests, d'autres le sont difficilement et restent tacites pour l'expert. L'idée est donc de présenter des données à l'expert, au fur et à mesure du processus de diagnostic pour lui permettre de juger en s'appuyant sur sa connaissance tacite. La problématique est donc de savoir ce qui doit être présenté à l'expert pour pouvoir exploiter l'expertise implicite interactivement.

L'interaction homme-automate pour exploiter l'expertise implicite est considéré dans ce chapitre en posant un cas d'étude étudié dans le cadre du projet Hydrodiag (Nguyen [2005]; Giap *et al.* [2010a]), mené en coopération avec Christian Depraetere, chercheur à l'Institut de Recherche pour le Développement. L'objectif était de diagnostiquer des défauts sur un réseau de pluviomètres. La description de ce cas d'étude ainsi que les outils utilisés pour résoudre

le problème seront détaillés plus tard. Résumons ici les étapes principales qui nous amènent au problème considéré. La première étape consiste à concevoir des tests de détection. Un ensemble de tests est généré : ils reposent sur des corrélations entre les quantités d'eau reçues par des capteurs hydrologiques. Les défauts multiples de capteurs sont courants dans ce type de problème. Pour faire face à cette difficulté, l'algorithme de l'arbre HS (Reiter [1987]) a été utilisé. Le nombre de diagnostics donnés par l'algorithme est considérable au point d'en devenir difficilement exploitable par un opérateur. Par exemple, dans le problème Hydrodiag, 16 différents diagnostics peuvent être trouvés avec 5 ou 6 défauts simultanés pour chaque diagnostic. Comme il n'est pas raisonnable de présenter l'ensemble des diagnostics calculés à un expert, imaginer des interactions homme-automate paraît judicieux pour localiser les défauts sans noyer l'expert dans une foultitude d'hypothèses qu'il ne sait plus analyser. Ceci pose le problème à résoudre de ce chapitre.

Ce chapitre propose des contributions dans le contexte d'interaction homme-machine dans l'analyse diagnostique en complétant les travaux présentés dans (Nguyen [2005]). Dans le cadre du projet Hydrodiag, (Nguyen [2005]) se focalise sur la conception des tests de détection, la génération de résidus et la conception d'un seuil de détection en logique nette. Les travaux de Nguyen, qui font la base de notre problème étudié, sont rappelés en détail dans la partie de cas d'étude, dans la section 5.2.1. Pour compléter les travaux de (Nguyen [2005]), nous présentons dans la section 5.2.1 la conception d'une matrice de diagnostic interactive qui permet d'exploiter la connaissance implicite de l'hydrologue dans la phase d'analyse diagnostique. Pour une contribution, les résultats du raisonnement diagnostique flou présenté dans (Touaf et Ploix [2004b]) sont intégrés dans le cadre du projet Hydrodiag afin de minimiser les non-détections d'un côté et d'exploiter la connaissance implicite de l'hydrologue d'autre côté. Ceci sera détaillé plus tard dans la section 5.3.

5.2 Interactions home-automate pour guider l'opérateur parmi un ensemble de possibilités

Dans cette partie, nous allons nous focaliser sur la recherche d'une méthode pour guider l'opérateur à partir d'un ensemble de possibilités. L'idée principale est qu'à partir d'un ensemble de diagnostics possibles donnés par l'automate, des diagnostics qui sont moins probables seront retirés au fur et à mesure grâce aux nouvelles observations collectées ou de la connaissance de l'expert. L'interaction homme-automate se manifeste donc dans la phase de diagnostic. Dans la communauté de DX, certains travaux sont proposés en se basant sur ce principe (De Kleer et Williams [1987]; Davis et Hamscher [1988]; De Kleer et Williams [1992]; Struss [1992]; Ploix *et al.* [2003]; H. Ressencourt [June 26-28, 2006]). Pourtant, l'interaction homme-automate est présentée de différentes façons. Ceci dépend de la nature du modèle du système étudié. Typiquement, nous détaillons ici les travaux présentés dans (Kleer et

Williams [1987]; De Kleer et Williams [1992]) qui concernent l'aspect d'interaction homme-automate dans l'analyse diagnostique avant de proposer une solution pour notre cas d'étude.

Dans (Kleer et Williams [1987]; De Kleer et Williams [1992]), les deux étapes de détection (collection des symptômes) et de diagnostic ne sont pas clairement distinguées dans les travaux de De Kleer. L'idée principale est toujours de calculer des candidats (aussi appelé diagnostic dans notre travail) à partir d'un ensemble de conflits (aussi appelé symptôme dans notre travail). L'interaction se manifeste au niveau des mesures pour enlever au fur et à mesure les candidats qui sont moins probables. L'objectif final du travail de De Kleer est de localiser le défaut avec un nombre de mesures minimal. Un processus d'interaction homme-automate, consciemment ou non, a été présenté dans la phase de diagnostic dans le travail de De Kleer (bien que l'aspect d'interaction n'est pas clairement préciser par l'auteur). L'interaction entre l'expert et l'automate a lieu au niveau des mesures de la façon suivante : l'automate propose des prochaines mesures au testeur en s'appuyant sur la probabilité des défauts et la théorie de l'Entropie ; le testeur fait des mesures et fournit les valeurs collectés à l'automate. Ceci donne un outil quasiment automatique et ne permet pas d'exploiter l'expertise implicite.

En comparaison avec les travaux de De Kleer, nous présentons dans cette partie le cas où l'interaction homme-automate permet d'exploiter l'expertise implicite au fur et à mesure dans la phase de diagnostic pour localiser les défauts à partir d'un ensemble de possibilités. Le principe du processus de diagnostic est illustré sur la figure 5.1.

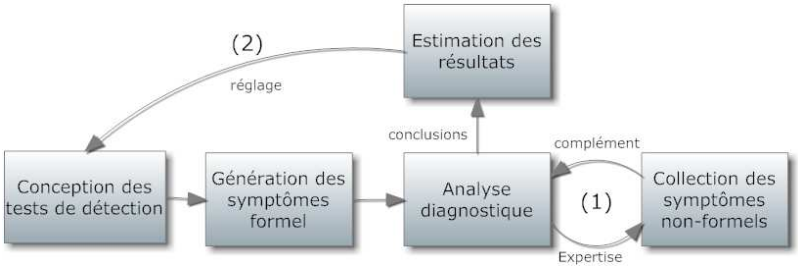


FIG. 5.1 – Processus de diagnostic interactif

La première étape est la modélisation du système qui aboutit à un ensemble de tests de détection. Les symptômes sont collectés automatiquement grâce aux tests de détection et aux observations. Soit $\mathbb{T}$ un ensemble de tests valides. Pour simplifier les notations, on utilisera une fonction vérité $\mu(P)$ à valeur dans $\{0,1\}$, où P est une proposition logique, définie par : $\bot(P) \leftrightarrow \mu(P) = 1$ et $\top(P) \leftrightarrow \mu(P) = 0$. Notons que, dans l'approche FDI (section 1.2.3, page 8), l'hypothèse d'exonération revient à considérer que si un test est consistant alors il n'y a aucun défaut dans le sous-système testé, les composants impliqués dans ce sous-système sont exonérés. Généralement, un test consiste à tester un ensemble de modes de comportement des éléments impliqués dans un sous-système testable. Nous allons considérer chaque test $test_i = (\mathscr{K}_i, \mathscr{K}_i') \in \mathbb{T}$ de deux façons différentes :

Sans hypothèse d'exonération, si un test est consistant, il n'est pas sûr que les modes testés

soient effectifs. Ceci est exprimé par :

$$\mu\left(\bigwedge mode_i \in Modes(test_i)\right) \leq \mu(test_i) \tag{5.2.1}$$

Avec hypothèse d'exonération, si un test est consistant alors les modes impliqués dans ce test (les modes testés)sont effectifs. Ceci est exprimé par :

$$\mu\left(\bigwedge mode_i \in Modes(test_i)\right) = \mu(test_i) \tag{5.2.2}$$

Dans la communauté bridge DX-FID, l'hypothèse d'exonération n'est pas prise en compte. Les diagnostics est donc calculés à partir des propositions logiques $\bigwedge mode_i \in Modes(test_i)$ où $\mu(test_i) = 0$. Une fois qu'un ensemble de diagnostics possibles est obtenu, la question posée est comment retirer au fur et à mesure les diagnostics qui sont les moins probables et localiser les défauts le plus rapidement possible. Nous parlons donc de l'interaction homme-automate dans la phase de diagnostic qui est présenté par (1) sur la figure 5.1. Pour la localisation de défauts sur un réseau de pluviomètres que nous avons évoquées dans la section 5.1, l'opérateur ne fait pas des mesures mais qu'il estime directement les défauts de capteurs en observant le hyétogramme de pluie des capteurs. Une rétro-analyse (2) est aussi considérée pour corriger les erreurs de seuil au niveau des tests automatiques. Durant l'interaction (1), le rôle de l'automate est de guider l'opérateur pour localiser les défauts le plus rapidement possible. Pour ce but, la mesure de coïncidence et la probabilité des défauts sont exploitées. Elles peuvent être consultées en détail dans l'annexe A de ce manuscrit.

En se basant sur ce principe, nous avons proposé dans le cadre du projet Hydrodiag la conception d'une matrice de diagnostic interactive pour exploiter l'expertise implicite et pour guider l'expert durant le processus de diagnostic.

5.2.1 Conception d'une matrice de diagnostic interactive

Pour résoudre le problème, nous nous sommes orientés vers une solution d'accompagnement de l'expert pour l'établissement d'un diagnostic. L'idée est que seule une partie de la connaissance de l'expert peut être formalisée sous la forme de tests automatiques. Il y a d'une part le système d'aide au diagnostic avec des modèles mathématiques précis, avec des outils de raisonnement exacts capables d'appréhender la complexité sans difficulté, et de l'autre, un expert avec une connaissance tacite complémentaire qui lui permet souvent de déterminer si un capteur est défaillant ou pas en regardant son hyétogramme et ceux de ses voisins alors que le système d'aide ne peut pas le détecter.

La procédure consiste à exploiter les résultats des tests automatiques et les modes de défaut préalablement sélectionnés par un expert, pour lui présenter des informations judicieuses qui lui permettront, en se basant sur une connaissance implicite, de sélectionner de nouveaux modes de défaut. Néanmoins, pour que cela fonctionne, il faut offrir une certaine liberté de

choix à l'expert car il est difficile de prédire où il aura le plus de facilité à conclure. Dans ce but, un outil d'aide au diagnostic est conçu pour guider l'expert durant le processus de diagnostic.

Avant de présenter l'outil d'aide au diagnostic, nous présentons en détail les travaux existant dans Nguyen [2005], c'est un cas d'étude intéressant pour le contexte interaction homme-automate.

Cas d'étude

Nous allons détailler maintenant comment se formule le problème de détection de défaut sur un réseau de pluviomètres. Le problème posé consiste à mettre au point un outil d'aide au diagnostic pour déterminer les défauts d'un réseau de capteurs répartis sur le bassin versant de l'Ouémé de $47536 km^2$ au Bénin. 46 pluviomètres à godets, qui basculent à chaque fois que $5mm$ d'eau ont été reçus, sont disponibles. Leurs coordonnées *latitude* et *longitude* sont connues. Les capteurs sont répartis sur le bassin de l'Ouémé conformément à la figure 5.2.

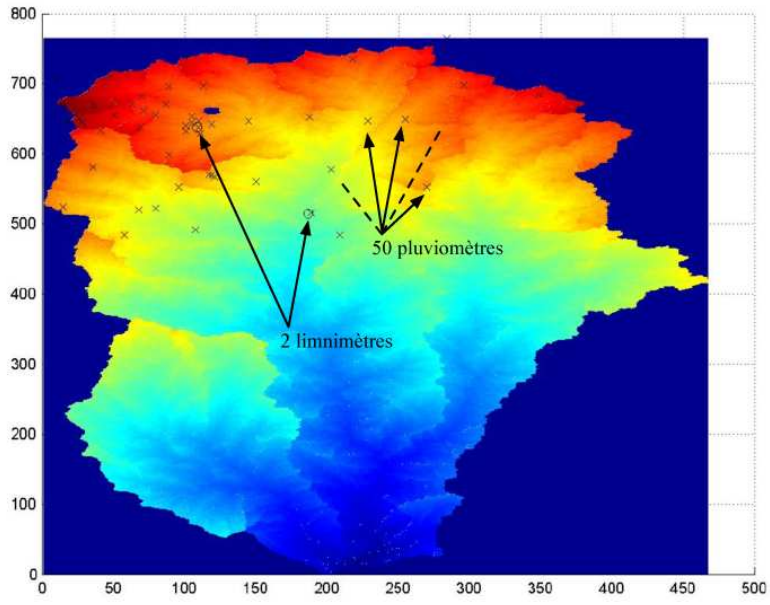


FIG. 5.2 – Observatoire OHHVO du bassin de l'Ouémé au bénin et réseau de capteurs

La première étape d'une procédure de diagnostic automatisée est généralement la conception de tests de détection. Pour tester les pluviomètres, un simple outil exploitant la redondance matérielle (comparaison de données de capteurs mesurant une grandeur proche) est utilisé en recherchant des pluviomètres corrélés deux à deux. A l'aide du retour d'expérience des hydrologues, les corrélations calculées au niveau de chaque mois des couples de capteurs séparés spatialement par une distance moins de $10km$ sont testés [1] : cela caractérise la validité du

[1] Les hydrologues considèrent que, pour les événements pluvieux considérés, deux capteurs distants de plus

modèle de corrélation. Du fait de la faible densité du réseau de capteurs, seuls 25 des 46 pluviomètres peuvent être testés. La figure 5.3 montre les tests qui peuvent être effectués.

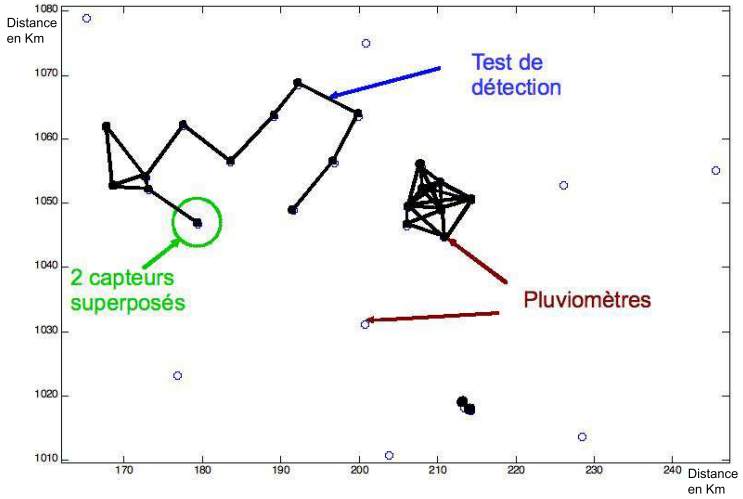


FIG. 5.3 – Position des capteurs et tests réalisés sur le bassin de l'Ouémé

Pour établir des seuils de décision au niveau des corrélations, nous disposons de tableaux identifiant l'état de chacun des capteurs pour chaque période de deux semaines (voir figure 5.4).

La figure 5.5 montre comment choisir un seuil sachant que la validité indique qu'il ne faut considérer que les capteurs distants de moins de 10km. Il faut d'une part minimiser le nombre de non détection tout en interdisant strictement les fausses alarmes qui sont critiques pour une analyse diagnostique logique car les diagnostics obtenus seraient erronés alors qu'une non-détection n'engendrerait que des diagnostics moins détaillés (certains modes de défaut pourraient ne pas apparaître). Plus concrètement, sur la figure 5.5, il faut qu'il existe un minimum de 'x' rouges, qui indiquent des résidus de tests de corrélation jugés inconsistants par un expert, en dessous du seuil et qu'il n'existe aucun '+' vert, qui indiquent des résidus de tests de corrélation jugés consistants par un expert, au dessus du seuil. Au passage, on constate que de nombreux états défaillants s'apparentent à des états normaux : tous les modes de défaut ne se manifestent pas en permanence. L'analyse de la figure 5.5 permet d'obtenir *une fonction seuil* linéaire en fonction de la distance telle que tous les points qui représentent des tests normaux sont au dessous de celle-ci. Cette *fonction seuil* est représentée par une droite sur la figure 5.5

Nous avons alors appliqué un algorithme d'analyse diagnostique en pensant être en passe de résoudre le problème. Or, pour les huit périodes mensuelles étudiées (de mars à octobre car en dehors de cette période, il ne pleut pas sur ce bassin), nous avons obtenu de 13 à 32 diagnostics possibles pour chacun des 8 mois de l'année 2002, sachant qu'un diagnostic comporte au minimum 4 capteurs simultanément défaillants. Par exemple, pour le mois d'août 2002, nous avons obtenu 16 diagnostics :

de 10*km* reçoivent des quantités de pluies indépendantes les unes des autres

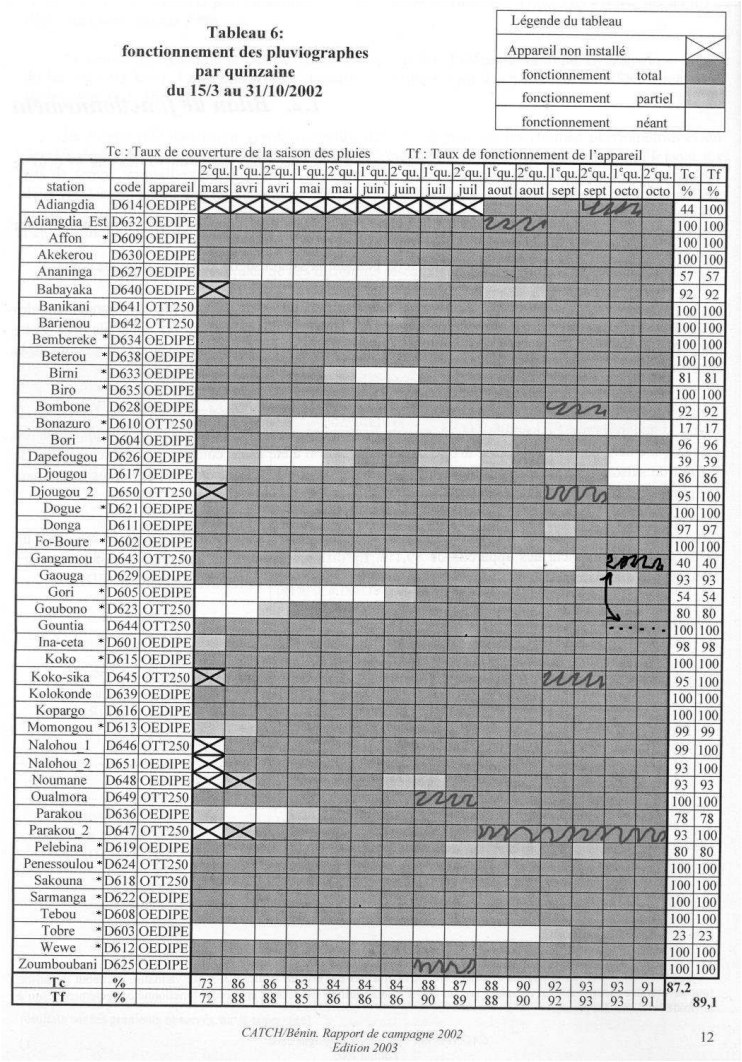


Tableau 6:
fonctionnement des pluviographes
par quinzaine
du 15/3 au 31/10/2002

Légende du tableau	
Appareil non installé	
fonctionnement total	
fonctionnement partiel	
fonctionnement néant	

Tc : Taux de couverture de la saison des pluies Tf : Taux de fonctionnement de l'appareil

station	code	appareil	2°qu. mars	1°qu. avri	2°qu. avri	1°qu. mai	2°qu. mai	1°qu. juin	2°qu. juin	1°qu. juil	2°qu. juil	1°qu. aout	2°qu. aout	1°qu. sept	2°qu. sept	1°qu. octo	2°qu. octo	Tc %	Tf %
Adiangdia	D614	OEDIPE																44	100
Adiangdia_Est	D632	OEDIPE																100	100
Affon *	D609	OEDIPE																100	100
Akekerou	D630	OEDIPE																100	100
Ananinga	D627	OEDIPE																57	57
Babayaka	D640	OEDIPE																92	92
Banikani	D641	OTT250																100	100
Barienou	D642	OTT250																100	100
Bembereke *	D634	OEDIPE																100	100
Beterou *	D638	OEDIPE																100	100
Birni *	D633	OEDIPE																81	81
Biro *	D635	OEDIPE																100	100
Bombone	D628	OEDIPE																92	92
Bonazuro *	D610	OTT250																17	17
Bori *	D604	OEDIPE																96	96
Dapefougou	D626	OEDIPE																39	39
Djougou	D617	OEDIPE																86	86
Djougou_2	D650	OTT250																95	100
Dogue *	D621	OEDIPE																100	100
Donga	D611	OEDIPE																97	97
Fo-Boure *	D602	OEDIPE																100	100
Gangamou	D643	OTT250																40	40
Gaouga	D629	OEDIPE																93	93
Gori *	D605	OEDIPE																54	54
Goubono *	D623	OTT250																80	80
Gountia	D644	OTT250																100	100
Ina-ceta *	D601	OEDIPE																98	98
Koko *	D615	OEDIPE																100	100
Koko-sika	D645	OTT250																95	100
Kolokonde	D639	OEDIPE																100	100
Kopargo	D616	OEDIPE																100	100
Momongou *	D613	OEDIPE																99	99
Nalohou_1	D646	OTT250																99	100
Nalohou_2	D651	OEDIPE																93	100
Noumane	D648	OEDIPE																93	93
Oualmora	D649	OTT250																100	100
Parakou	D636	OEDIPE																78	78
Parakou_2	D647	OTT250																93	100
Pelebina *	D619	OEDIPE																80	80
Penessoulou *	D624	OTT250																100	100
Sakouna *	D618	OTT250																100	100
Sarmanga *	D622	OEDIPE																100	100
Tebou *	D608	OEDIPE																100	100
Tobre *	D603	OEDIPE																23	23
Wewe *	D612	OEDIPE																100	100
Zoumboubani	D625	OEDIPE																100	100
Tc	**%**		73	86	86	83	84	84	84	88	87	88	90	92	93	93	91	87.2	
Tf	**%**		72	88	88	85	86	86	86	90	89	88	90	92	93	93	91		89,1

CATCH/Bénin. Rapport de campagne 2002
Edition 2003

12

FIG. 5.4 – Une partie de l'expertise sur le fonctionnement des pluviomètres

```
diagnosis #0 (Formal:100.0%, Contextual:88.63636363636364%, A priori:0.010000000000000004%)
Component d643 (gangamou) is faulty
 and component d626 (dapefougou) is faulty
 and component d614 (adiangdia) is faulty
 and component d647 (parakou_2) is faulty
------------------------------
diagnosis #1 (Formal:100.0%, Contextual:88.63636363636364%, A priori:0.010000000000000004%)
Component d643 (gangamou) is faulty
 and component d626 (dapefougou) is faulty
 and component d632 (adiangdia_oues) is faulty
 and component d647 (parakou_2) is faulty
------------------------------
...
------------------------------
diagnosis #15 (Formal:100.0%, Contextual:72.72727272727273%, A priori:1.0000000000000003E-4%)
```

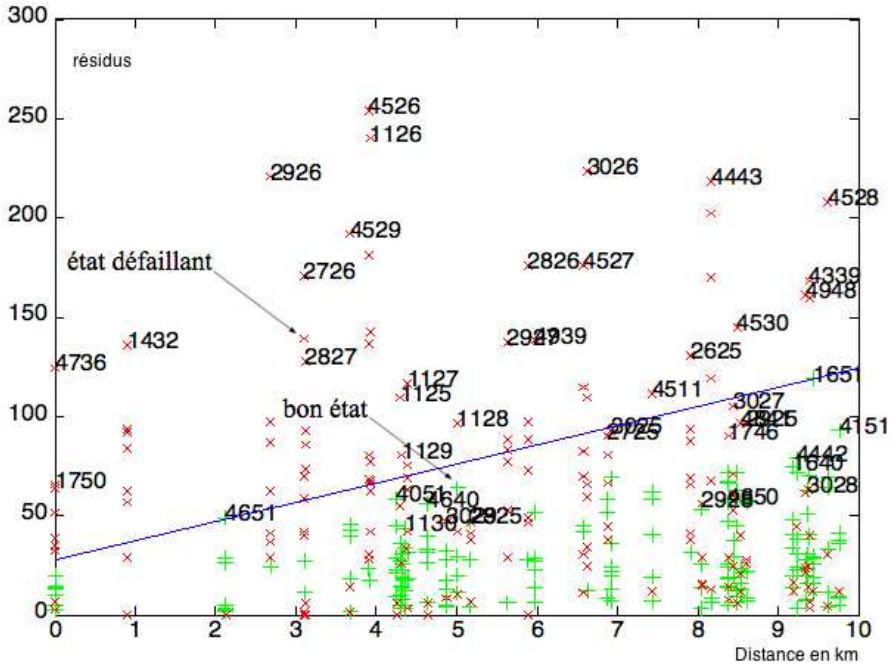


FIG. 5.5 – Seuils de détection

```
Component d639 (kolokonde) is faulty
 and component d611 (donga) is faulty
 and component d644 (gountia) is faulty
 and component d645 (koko-sika) is faulty
 and component d632 (adiangdia_oues) is faulty
 and component d636 (parakou) is faulty
```

Ces résultats induisent de nombreuses questions : comment un expert est-il capable d'exploiter de tels résultats ? Dans quelle direction partir ? Peut-on réellement parler d'aide au diagnostic ici ? Plutôt que d'essayer de déterminer lequel de ces diagnostics est correct, n'est-il pas plus simple pour un expert d'analyser directement les 25 hyétogrammes comme il le faisait jusque là ?

Pour proposer un outil d'aide au diagnostic utile à ce type de contextes, il va falloir s'appuyer sur un nouveau paradigme que nous avons déjà étudié à travers d'autres types de problèmes : l'aide au diagnostic interactive.

5.2.2 Matrice de diagnostic interactive

Pour éviter la présentation d'un ensemble de diagnostics donné par l'algorithme basé sur HS-Tree, nous nous sommes orientés vers une solution d'accompagnement de l'expert pour l'établissement d'un diagnostic. Nous avons conçu une *matrice de diagnostic interactive*, pré-

sentée par la figure 5.6.

La matrice de diagnostic interactive s'appuie sur les résultats des tests $\mu(test_i)$ avec $test_i \in \mathbb{T}$ et $\mathbb{T}$ est un ensemble de tests valides. Commençons par examiner la zone (A) de la figure 5.6. En (B), on trouve la liste des items constituant le diagnostic en construction D_j établi à l'itération i par l'expert sachant que le bouton (H) permet à tout moment de retirer le dernier item ajouté au diagnostic en construction pour se réorienter vers une nouvelle direction explicative.

La zone (C) présente des indicateurs caractérisant la pertinence du diagnostic en construction. S'il existe au moins un test $test_i$ totalement insatisfait ($\mu(test_i) = 0$), alors un diagnostic peut être calculé à partir des tests $\{test_i \in \mathbb{T}; \mu(test_i) = 0\}$ et le degré de vérité pour les diagnostics obtenus, noté *FORMAL* ou F dans la matrice de diagnostic interactive, vaut 1 (voir Eq. (.0.11)), sinon il vaut 0. Ainsi, l'indicateur *FORMAL* est fixé à 100% si le diagnostic en construction contient des modes de défauts expliquant tous les tests totalement insatisfaits. Dans le cas contraire, l'indicateur *FORMAL* vaudra 0% : les anomalies sûres constatées ne sont pas encore expliquées d'un point de vue formel. Les cases *explication* dans la colonne (E) signifient, si elles sont cochées, que l'item de la ligne correspondante explique des tests totalement insatisfaits non encore expliqués.

L'indicateur *CONTEXTUAL*, noté aussi C, représente la mesure de coïncidence présentée dans l'annexe A. Soit $T = (t_i)$ une liste ordonnée des tests, et $D = d_i$ un ensemble de diagnostics. La mesure de coïncidence est donnée par :

$$\forall d_i \in D, \mu_T^c(d_i) = \frac{|\sigma_T(d_i), \sigma_T^*|_{Hamming}}{card(T)} \tag{5.2.3}$$

Pour se ramener sur une échelle où 100% est préférable, l'indicateur *CONTEXTUAL* est défini par : $contextual_{\mathbb{T}}(D_j) = 1 - \mu_T^c(d_i)$.

L'indicateur *A PRIORI*, noté aussi A, s'évalue à partir des probabilités d'occurrence a priori des modes de défauts (voir l'annexe A) : chaque probabilité $p(cfm\ (item_i))$ est définie pour tout item $item_i$. Pour un diagnostic D_j, l'indicateur *A PRIORI* vaut : $\Pi_{cfm\ (item_i) \in D_j} p(cfm\ (item_i))$. La fonction log est utilisée dans la zone (F) pour faciliter la lecture.

La colonne (D) de la matrice contient les modes de défaut qui, s'ils étaient sélectionnés par l'expert, permettraient d'expliquer certains des tests insatisfaits non encore expliqué par le diagnostic en construction. Soient $\mathbb{T}^{insatisfait}_{expliqués}(D_j) \in \mathbb{T}^{insatisfait}$, l'ensemble des tests de $\mathbb{T}^{insatisfait}$ qui vérifient : $\{test_i \in \mathbb{T}^{insatisfait}; Expl(test_i) \cap D_j \neq \emptyset\}$, et son complémentaire $\mathbb{T}^{insatisfait}_{inexpliqués}(D_j)$ dans $\mathbb{T}^{insatisfait}$. D_j doit être complété afin de vider $\mathbb{T}^{insatisfait}_{inexpliqués}(D_j)$. Les items candidats affichés dans la colonne (D) sont donc donnés par $\bigcup_{test_i \in \mathbb{T}^{insatisfait}_{inexpliqués}(D_j)} Expl(test_i)$.

Les zones (F) et (G) correspondent aux indicateurs de pertinence (C) et aux suggestions (D) qui seraient obtenues en ajoutant l'item de la ligne concernée au diagnostic en construction.

Le bouton *Exonération* permet de retirer définitivement certains items des suggestions (colonne (D)) pour la construction d'un diagnostic.

Enfin, pour aider les experts, les hauts des colonnes (E) et (F) peuvent être cliqués pour modifier le classement des items suggérés.

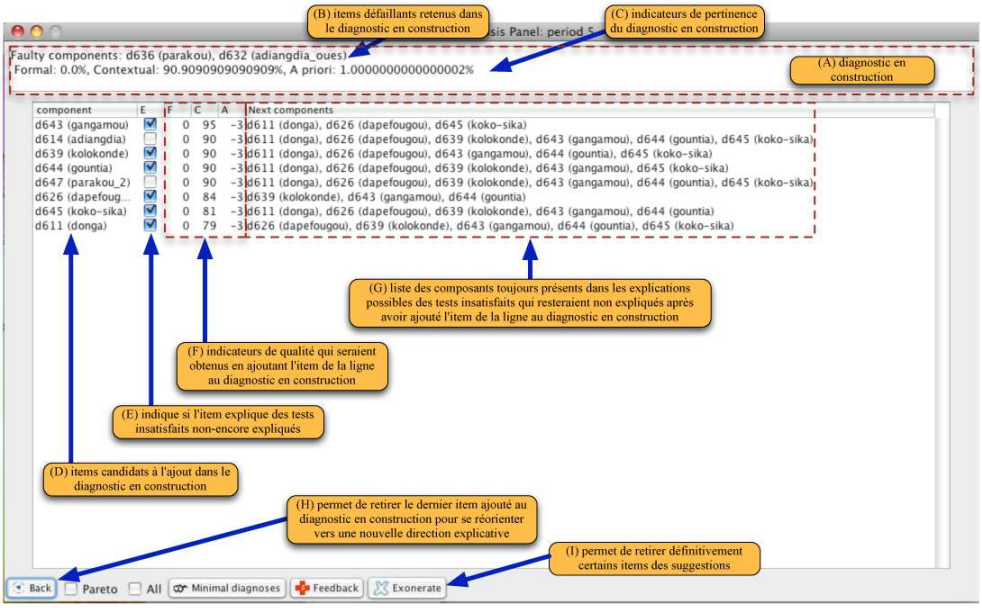


FIG. 5.6 – Matrice de diagnostic interactive

5.2.3 La rétro-analyse

Au cours du processus de diagnostic, le système de diagnostic développé peut conduire à des résultats erronés. Nous appellerons rétro-analyse l'opération qui consiste à analyser les performances des tests de détection lorsque l'état réel du système diagnostiqué est fourni par l'expert. Il s'agit notamment d'identifier les tests ayant induits des diagnostics erronés. En effet, quand une fausse-alarme est détectée, une intervention de l'expert paraît judicieuse pour ajuster certains tests, en recalibrant des seuils de détection par exemple. Supposons que les résultats des tests valides soient connus et que le diagnostic effectif, noté D^*, constaté par un opérateur soit lui aussi connu. Nous proposons l'algorithme qui suit pour résoudre le problème.

D^* : **un ensemble de modes correspondant à l'état réel du système**
$tests$: **un ensemble de tests effectués sur le système**
Pour $test \in tests$ **faire**
  Si ($test$ est valide et non satisfait) **Alors**
    Si ($Expl(test) \cap D^* = \emptyset$) **Alors**
      [fausse alarme critique : test est à revoir]
    Fin Si
  Fin Si
  Si ($test$ est valide et satisfait) **Alors**
    Si ($Expl(test) \cap D^* \neq \emptyset$) **Alors**
      [non détection : test n'a pas réagi au défaut courant]
    Fin Si
  Fin Si
Fin Pour

Il faut noter qu'une fausse-alarme est critique car il n'est pas acceptable qu'un test révèle une anomalie alors qu'il n'y en a pas : le test doit être recalibré, en remontant par exemple les seuils de détection ou les rayons des intervalles modélisant les incertitudes, voir être reconçu si nécessaire.

Revenons sur le système de cuve. Imaginons qu'un opérateur constate que $ensemble2$ est défaillant, c'est-à-dire $D^* = \{cfm\ (ensemble2)\}$. Aucun des diagnostics calculés ne correspond. Une rétro-analyse est donc justifiée.

Pour mémoire, les 3 tests sont valides est seul $test_3$ n'est pas satisfait. Les modes associés aux tests sont donnés par la table de signature 10. En l'occurrence, on a :

$$Expl(test_1) = \{cfm\ (bac1), cfm\ (vanne1), cfm\ (ensemble2)\}$$
$$Expl(test_2) = \{cfm\ (bac1), cfm\ (capteur1), cfm\ (ensemble2)\}$$
$$Expl(test_3) = \{cfm\ (bac1), cfm\ (capteur1), cfm\ (vanne1)\}$$

En appliquant l'algorithme précédent, on obtient le résultat suivant :
– fausse alarme critique : $test3$ est à revoir
– $test1$ n'a pas détecté le défaut
– $test2$ n'a pas détecté le défaut

Le fait que $test1$ et $test2$ n'aient pas détecté le défaut n'est pas forcément un problème : il se peut que les observations effectuées ne permettent pas de mettre en évidence l'anomalie. En revanche, la fausse alarme au niveau de $test3$ doit être prise très au sérieux car rien ne peut

expliquer que, si l'état réel du système est effectivement *cfm* (*ensemble*2), *test*3 révèle une anomalie. Il faudrait revoir *test*3 pour éviter que la situation ne se reproduise.

5.2.4 Application

Examinons l'utilisation de cette matrice interactive sur un scénario correspondant à la recherche des pluviomètres défaillants durant le mois d'août 2002. L'expert découvre la figure 5.7. Comme l'indicateur *CONTEXTUAL* est plus important pour d643 (d643 explique beaucoup de symptômes), l'expert choisit naturellement ce diagnostic. La position du capteur s'allume alors en bleu sur la carte du logiciel *hydrodiag* et une fenêtre avec les hyétogrammes des capteurs situés dans un rayon de 10km apparaît (voir figure 5.8). Visiblement, le capteur d643 est défaillant. L'expert poursuit cette direction explicative.

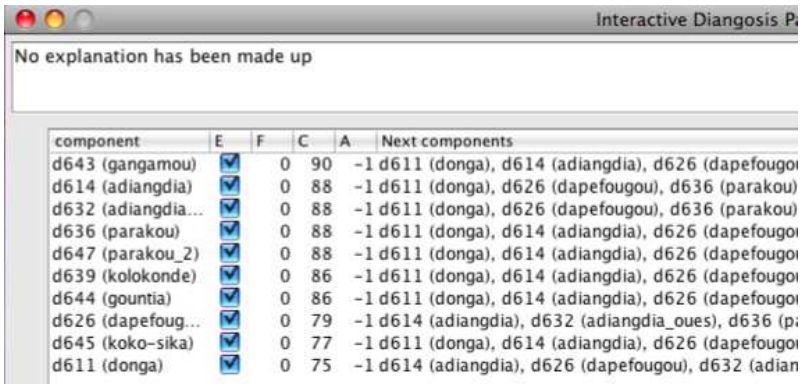


FIG. 5.7 – Analyse diagnostique pour le mois d'août 2002 - Étape 1

Quatre capteurs apparaissent alors en tête avec des scores comparables. L'expert choisit, au hasard ou selon son intuition, le premier (voir figure 5.9) : le d614. En regardant le hyétogramme du capteur et de son unique voisin, il en déduit que c'est le capteur d632 qui est défaillant et non d614. Il revient en arrière en cliquant sur le bouton *Back* (H) et se réoriente vers d632. Il obtient alors la matrice de la figure 5.10.

L'expert choisit alors d636. Le résultat obtenu confirme que cette direction est bonne (voir figure 5.11).

Il apparaît alors que le capteur d626 peut expliquer à lui seul les tests totalement insatisfaits restants. L'expert le sélectionne et vérifie la direction explicative sur le hyétogramme (voir figure 5.12). La direction s'avère être bonne et tous les tests totalement insatisfaits sont expliqués. Les pluviomètres défaillants sont donc : d643 (gangamou), d632 (adiangdia-oues), d636 (parakou), d626 (dapefougou). En se souvenant de la liste des 16 diagnostics possibles avec 4 défauts simultanés au minimum, on comprend aisément l'intérêt d'une approche interactive. Le diagnostic trouvé correspond au quatrième de la liste des diagnostics possibles calculés.

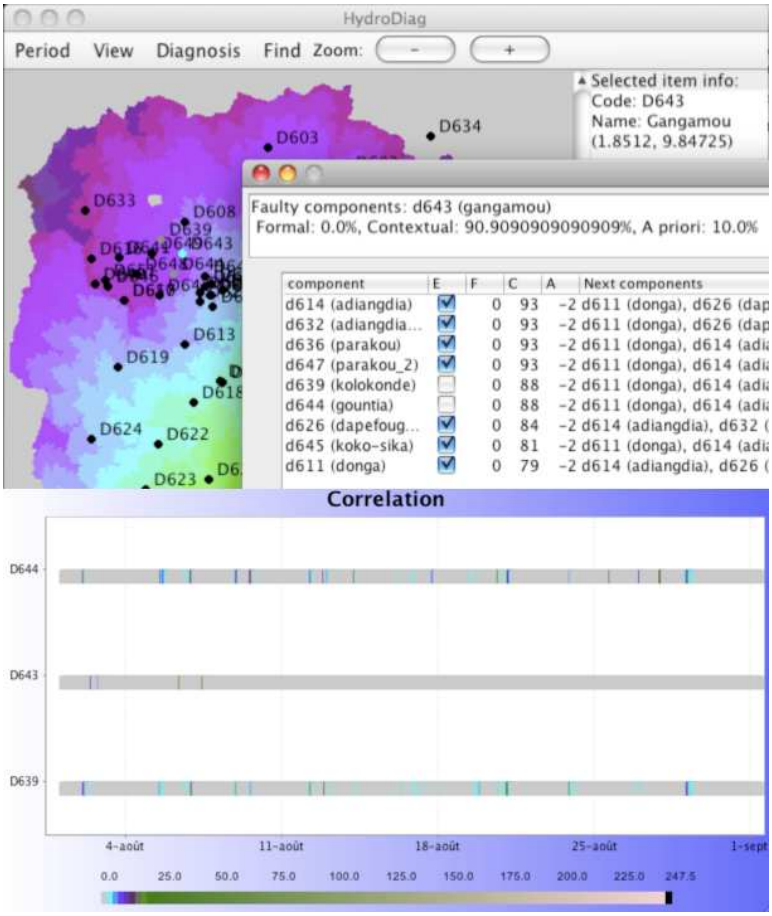


FIG. 5.8 – Analyse diagnostique pour le mois d'août 2002 - Étape 2

Grâce à cette approche, même des non-experts hydrologues peuvent trouver, la plupart du temps, les pluviomètres défaillants à tel point que nous avons trouvé des capteurs défaillants qui ne nous avaient pas été signalés par l'expert hydrologue et qui, après réexamen, les indique comme tels.

5.3 Prise en compte du doute : le raisonnement diagnostique flou

Vu l'impact d'une fausse alarme, il peut être difficile de régler les seuils de détection d'un test. En effet, pour éviter qu'elle ne se produise, le réglage peut devenir très pessimiste et engendrer ainsi beaucoup de non détection.

Par ailleurs, les hydrologues considèrent ici qu'un test n'est valide que si les deux pluviomètres impliqués sont distants de moins de 10km. La décision valide/invalide doit-elle aussi nette ici ? A 9,9km, le test est valide et à 10,1km, il ne l'est plus.

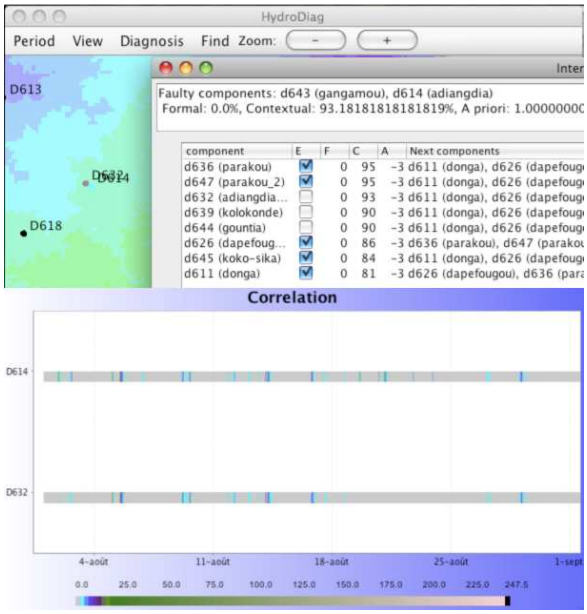


FIG. 5.9 – Analyse diagnostique pour le mois d'août 2002 - Étape 3

En plus des travaux présentés dans Nguyen [2005], nous proposons ici une contribution en exploitant la logique floue dans le cadre du projet Hydrodiag. Nous proposons tout d'abord dans la section 5.3.3 la conception d'un seuil de détection floue. En intégrant la logique floue dans dans la phase de détection, nous souhaitons éviter les fausses alarmes et minimiser au maximun possible les non-détections. L'expertise implicite est alors exploitée dans le processus de diagnostic pour localiser des défauts. Cette partie sera détaillée dans la section 5.3.3.

5.3.1 Raisonnement diagnostique flou

Dans le domaine d'analyse diagnostique, la logique floue est exploitée afin de faire face à des facteurs incertains. Certains travaux existant dans la littérature ont exploité la logique floue pour résoudre des problèmes de différentes natures, par exemple, le diagnostic médical, le diagnostic pour les processus chimiques, le diagnostic pour les systèmes industriels ...). Le problème a été résolu en associant au problème de diagnostic étudié dans le projet Hydrodiag, l'approche de diagnostic bridge FDI-DX que nous avons utilisée pour résoudre le problème avec les résultats du raisonnement diagnostique floue présenté dans (Touaf et Ploix [2004a,b]; Touaf [2005]) (voir l'annexe D). Dans ces travaux, les auteurs cherchent à transposer la logique nette de l'analyse diagnostique formelle à une logique floue de Zadeh [1965, 1975]. L'approche proposée permet d'éviter d'utiliser des mesures de nécessité Cayrac *et al.* [1996] et préserve les résultats de la logique nette lorsque les degrés d'appartenance deviennent certains 0 ou 1. Nous nous sommes appuyés sur l'interprétation de la logique floue proposée par Yager [1986]. La fonction de vérité $\mu(.)$ qui était à valeur dans $\{0,1\}$ en logique nette prend désormais valeur dans $[0,1]$: il s'agit d'un degré d'appartenance à vrai au sens de la logique

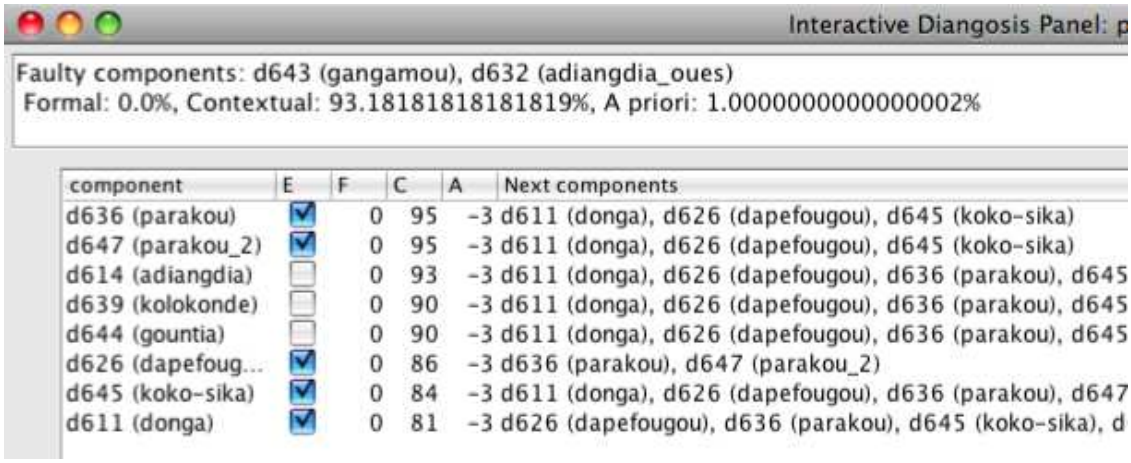


FIG. 5.10 – Analyse diagnostic pour le mois d'août 2002 - Étape 4

floue.

Dans la suite, nous allons rappeler les résultats de Touaf et Ploix [2004b] de façon à ce que nous puissions montrer comment les appliquer dans le cas du projet Hydrodiag. Le raisonnement détaillé de l'analyse diagnostique dans le contexte floue peut être consulté dans l'annexe C.

Fuzzification des symptômes

Soient $\mathscr{K}(V)$ et $\mathscr{K}'(V)$ respectiement le contrainte de comportement et le contrainte de validité modélisant le mode $ok(item_i)$. En phase de test, il existe deux degrés d'appartenance flou : $\mu_{\mathscr{K}} = \mu(\mathscr{K}(V) = 0)$ et $\mu_{\mathscr{K}'} = \mu(\mathscr{K}'(V) = 0)$. $\mu_{\mathscr{K}}$ et $\mu_{\mathscr{K}'}$ prennent des valeurs de 1 (satisfait) à 0 (non satisfait). Pour chaque test, une fonction de fuzzification permet d'évaluer le degré de vérité des symptômes en fonction des degrés de satisfaction des contraintes de comportement et de validité :

$$\mu(test) = \mu\left(\bigwedge_i ok(item_i)\right) = \Gamma(\mu_{\mathscr{K}}, \mu_{\mathscr{K}'}) = \frac{1 + (2\mu_{\mathscr{K}} - 1)\mu_{\mathscr{K}'}}{2} \tag{5.3.1}$$

En cas d'anomalie, au lieu d'évaluer $\mu\left(\bigwedge_i ok(item_i)\right)$, on va plutôt chercher à évaluer $\mu\left(\bigvee_i cfm\ (item_i)\right) = 1 - \mu\left(\bigwedge_i ok(item_i)\right) = 1 - \Gamma(\mu_{\mathscr{K}}, \mu_{\mathscr{K}'})$

Raisonnement diagnostique flou

L'analyse diagnostique peut être réalisé à partir du degré de vérité $\mu_{test} = \mu\left(\bigwedge_i ok(item_i)\right)$ de chaque test. Soit $\mathbb{T}$, l'ensemble des tests que l'on partitionne en $\mathbb{T} = \mathbb{T}^{satisfait} \cup \mathbb{T}^{insatisfait} \cup$

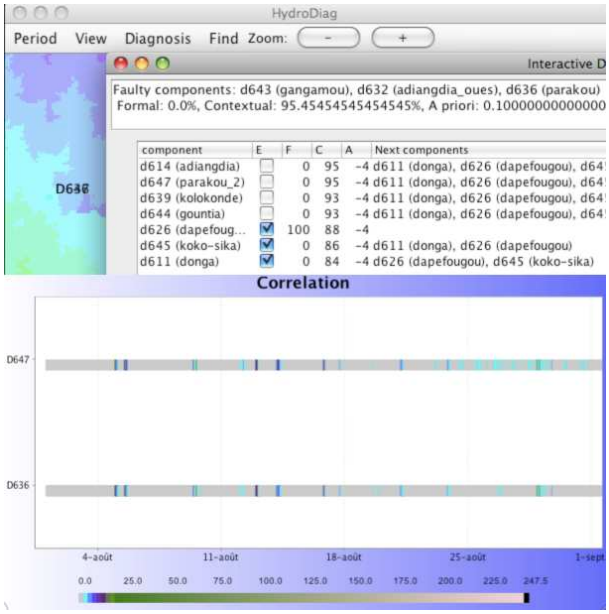


FIG. 5.11 – Analyse diagnostic pour le mois d'août 2002 - Étape 5

$\mathbb{T}^{douteux}$ avec $\mathbb{T}^{satisfait} = \{test; \mu(test) = 1\}$, $\mathbb{T}^{insatisfait} = \{test; \mu(test) = 0\}$ et $\mathbb{T}^{douteux} = \{test; \mu(test) \in]0,1[\}$, les trois cas suivants sont distingués :

- $\mathbb{T} \equiv \mathbb{T}_{satisfait}$: cette situation est similaire à celle de la logique nette : il n'y a aucune raison de calculer des diagnostics car aucune anomalie n'a été révélée.
- $\mathbb{T}^{insatisfait} \neq \emptyset$: les diagnostics doivent être calculés uniquement à partir des tests vérifiant $\mu(test) = 0$ et leur degré de vérité est de 1.
- $\mathbb{T}^{insatisfait} = \emptyset$ et $\mathbb{T}^{douteux} \neq \emptyset$: les diagnostics correspondent à des modes *cfm* simples : $\mathbb{D}^{douteux} = \bigcup_{test \in \mathbb{T}^{douteux}} \bigcup_{mode_i \in Expl(test)} mode_i$. Soient $cfm\ (item_i) \in \mathbb{D}^{douteux}$ et $\mathbb{T}^{douteux}$ un ensemble de tests douteux qui le contiennent. En utilisant *max* comme opérateur de fuzzification pour $\vee$, son degré de vérité sera donc :

$$\mu(cfm\ (item_i)) = max_{test \in \mathbb{T}^{douteux}} \left(1 - \Gamma(\mu_{\mathscr{K}_{test}}, \mu_{\mathscr{K}'_{test}})\right) \tag{5.3.2}$$

Pour classer les diagnostics, la distance Hamming est exploité dans le contexte floue. Une distance entre la signature effective définie par $\forall test_i \in \mathbb{T}, (\sigma^*_{\mathbb{T}})_i = 1 - \Gamma(\mu_{\mathscr{K}_{test_i}}, \mu_{\mathscr{K}'_{test_i}})$ [2], et la signature théorique d'un diagnostic. Soit un diagnostic D_j. La signature théorique de $\sigma_{\mathbb{T}}(D_j)$ est donnée par :

$$\forall test_i \in \mathbb{T}, \begin{cases} (\sigma_{\mathbb{T}}(D_j))_i = 0, & \text{si } Modes(D_j) \cap Expl(test_i) = \emptyset \\ (\sigma_{\mathbb{T}}(D_j))_i = 1, & \text{si } Modes(D_j) \cap Expl(test_i) \neq \emptyset \end{cases} \tag{5.3.3}$$

[2] s'il y a plusieurs jeux d'observations, $\sigma^*_{\mathbb{T}}$ correspond au maximum de $1 - \Gamma(\mu_{\mathscr{K}_{test_i}}, \mu_{\mathscr{K}'_{test_i}})$ sur tous les jeux

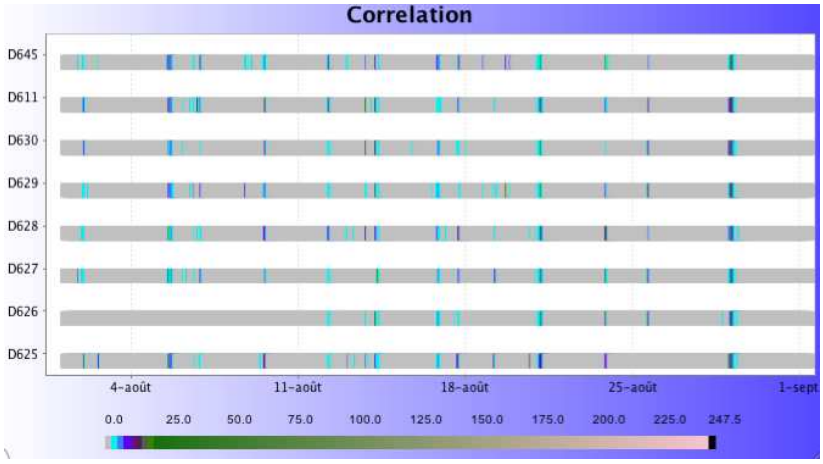


FIG. 5.12 – Analyse diagnostique pour le mois d'août 2002 - Étape 6

La distance entre les deux signatures s'écrit alors :

$$\forall d_i \in D, \mu_T^c(d_i) = \frac{|\sigma_T(d_i), \sigma_T^*|_1}{card(T)} \tag{5.3.4}$$

où :
- $(\sigma_{\mathbb{T}}^*)_i = \mu(test_i)$
- $\sigma_{\mathbb{T}}(D_j))$ est évalué comme indiqué précédemment
- une norme 1 [3] a été choisie pour resté cohérent avec la distance de Hamming de la logique nette (voir annexe A).

5.3.2 Application du raisonnement flou dans le problème Hydrodiag

Revenons sur le problème HYDRODIAG et examinons ce qu'apporte la logique floue. Une première approche consiste à comparer les degrés de vérité ou de satisfaction des tests en fonction des résidus constatés et des distances séparant les deux pluviomètres de chacun des tests. La figure 5.13 illustre les résultats obtenus. Pour obtenir ces graphiques, nous avons procédé par optimisation sous contrainte.

Rappelons que pour la logique nette, la contrainte et l'objectif du choix du seuil de détection sont :
- contrainte : pas de fausse-alarme, c'est-à-dire que tous les tests constatés consistants par l'expert, doivent avoir un résidu en dessous du seuil. Ce seuil avait été choisi comme une fonction linéaire de la distance. Tous les tests dont la distance des pluviomètres est supérieure à $d_{max} = 10km$ ne sont pas pris en compte.

[3] somme des valeurs absolues

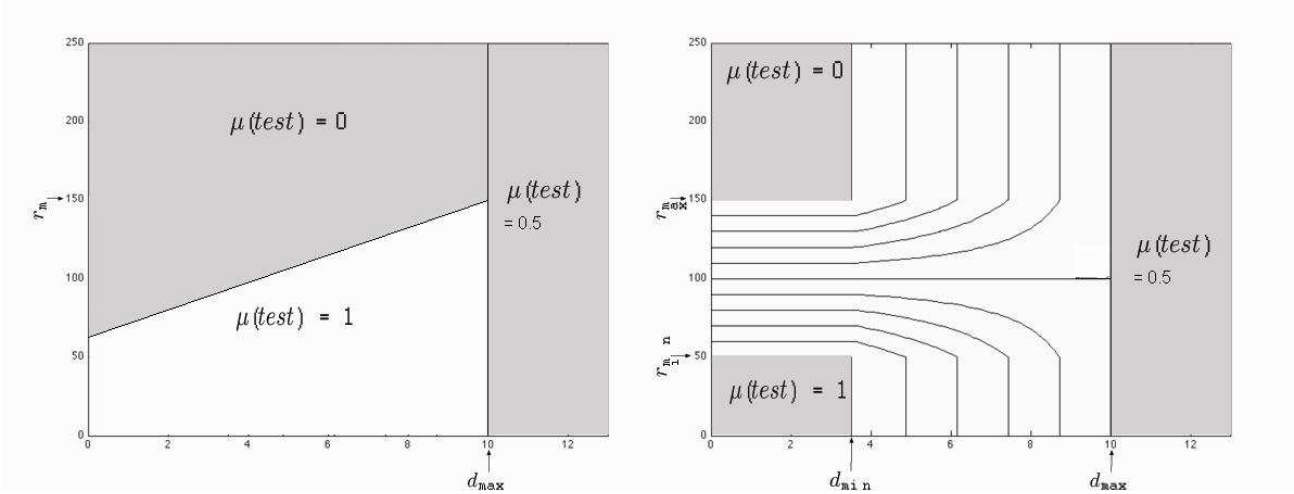


FIG. 5.13 – Degré de satisfaction d'un test en contexte net et flou

– objectif : recherche de *la fonction de seuil* qui minimise le nombre de non-détections, c'est-à-dire le nombre de tests constatés inconsistants par l'expert qui sont au-dessus de ce seuil linéaire. La forme d'un seuil linéaire est présentée sur la figure 5.5, page 138.

Pour la logique floue, nous allons déterminer tout d'abord les degrés de satisfaction des contraintes de comportement $\mu_{\mathscr{K}}$ et celui des contraintes de validité $\mu_{\mathscr{K}'}$ et puis nous calculons μ_{test}. A une distance d donnée entre deux capteurs, le degrée de satisfaction d'un test de corrélation $\mu_{\mathscr{K}}$ dépend de deux valeurs de résidus r_{min} et r_{max} (figure 5.14). En dessous de r_{min}, $\mu_{\mathscr{K}} = 1$ et au dessus de r_{max}, $\mu_{\mathscr{K}} = 0$. Pour la transition entre r_{min} et r_{max}, $\mu_{\mathscr{K}}$ a la forme linéaire :

$$\mu_K = \frac{r - r_{max}}{r_{min} - r_{max}} \tag{5.3.5}$$

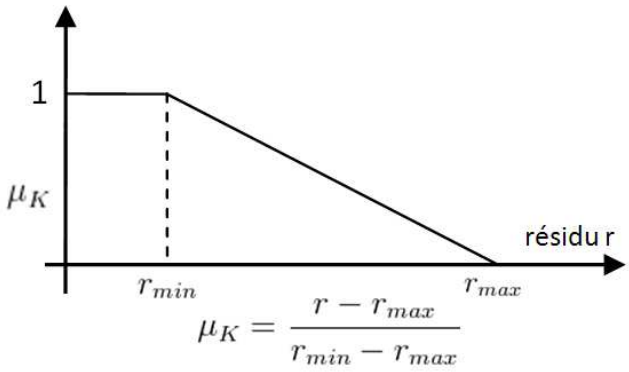


FIG. 5.14 – Degré de satisfaction de μ_K

Le degrée de satisfaction de la contrainte de validité $\mu_{\mathscr{K}'}$ dépend de la distance du couple de pluviomètres à tester d_{min} et d_{max} (figure 5.15). d_{max} reste fixé à $10km$ pour rester conforme

aux dires de l'expert. En dessus de d_{max}, $\mu_{\mathscr{K}'} = 0$. En dessous de d_{min}, $\mu_{\mathscr{K}'} = 1$. Pour la transition entre d_{min} et d_{max}, $\mu_{\mathscr{K}'}$ a la forme linéaire :

$$\mu_{K'} = \frac{d - d_{max}}{d_{min} - d_{max}} \tag{5.3.6}$$

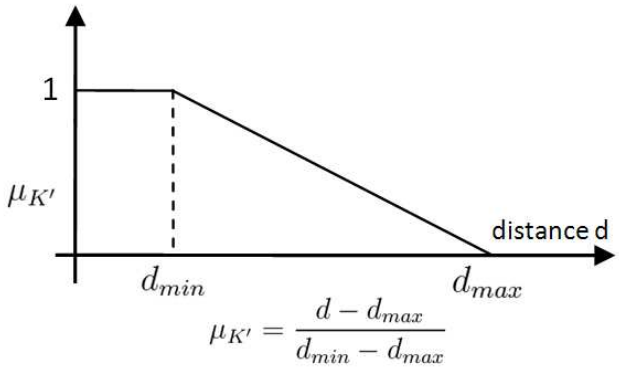


FIG. 5.15 – Degré de satisfaction de $\mu_{K'}$

Quand la logique floue est utilisée, il n'y a plus de paramètres à régler : r_{min}, r_{max} et d_{min} (d_{max} reste fixé à $10km$). Les contraintes à respecter et l'objectif d'optimisation sont :

– contraintes : pas de fausse-alarme. C'est-à-dire que le test est vérifié *ok* où $\mu(test) > 0$ ou

$$(1 - 2\mu(\mathscr{K}(test)))\mu(\mathscr{K}'(test)) < 1 \tag{5.3.7}$$

– objectif : minimisation des non-détections. Cela se traduit par le critère à minimiser suivant :

$$\underset{r_{min}, r_{max}, d_{min}}{min} \sum_{test \in \mathbb{T}_{inconsistants}} \mu(test) \tag{5.3.8}$$

où $\mathbb{T}_{inconsistants}$ représente l'ensemble des tests constatés inconsistants par le retour d'expérience de l'expert.

Le degré de satisfaction d'un test μ_{test} est calculé par l'équation .0.9 (page 193) à partir de $\mu_{\mathscr{K}}$ et $\mu_{\mathscr{K}'}$. Les valeurs de μ_{test} sont présentées sur la figure 5.16. En remplaçant $\mu_{\mathscr{K}}$ et $\mu_{\mathscr{K}'}$ dans μ_{test}, le calcul de μ_{test} qui correspond aux différentes zones de (r,d) est résumé dans le tableau 5.1. Les courbes r(d μ_{test}) sont donc trouvées, elles sont présentées sur la figure 5.16.

On constate une plus grande richesse en logique floue. Il est en effet possible de retrouver la logique nette si r_{min} se confond avec r_{max} et si d_{min} se confond avec d_{max} (figure 5.13).

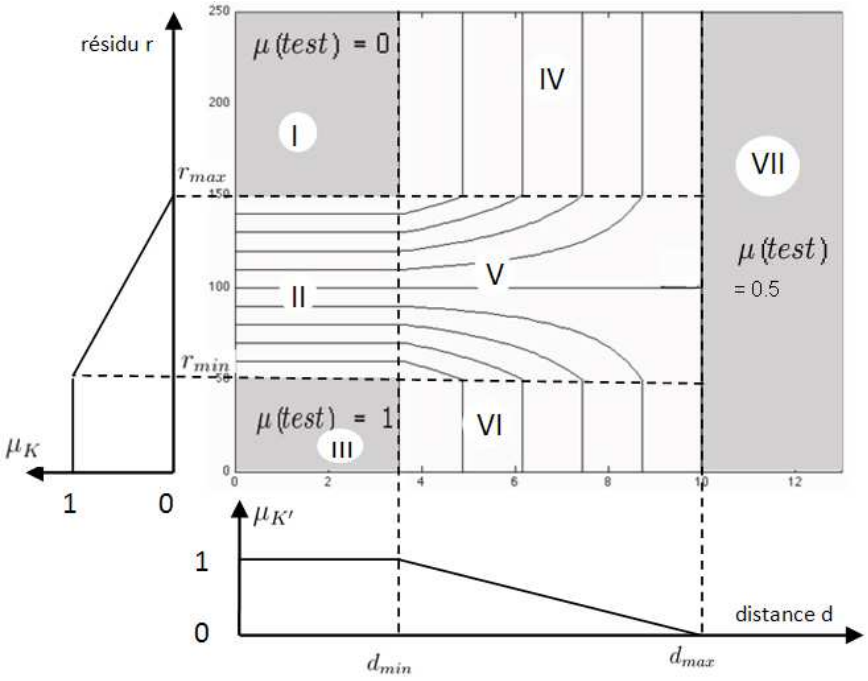


FIG. 5.16 – Degré de satisfaction d'un test en contexte flou

Interactions homme-automate dans un contexte flou

Suite de l'étape de détection (conception d'un seuil de détection flou), nous passons à l'étape de localisation des défauts pour le problème de hydrodiag. En nous appuyant sur les principes du raisonnement diagnostique flou ayant été posés dans la section 5.3.1, nous voyons maintenant comment intégrer ces principes dans une approche itérative et interactive du diagnostic. Revenons donc à la matrice de diagnostic interactive de la figure 5.6 et examinons les changements induits.

Au sein de la zone (A), la zone (C) présente des indicateurs caractérisant la pertinence du diagnostic en construction. Nous avons vu dans le paragraphe 6 que s'il existe au moins un test $test_i$ totalement insatisfait ($\mu(test_i) = 0$), alors le diagnostic est calculé comme on le ferait en logique nette uniquement à partir des tests $\{test_i \in \mathbb{T}; \mu(test_i) = 0\}$ et le degré de vérité pour les diagnostics obtenus, noté *FORMAL* ou F dans la matrice de diagnostic interactive, vaut 1 (voir Eq. .0.11). S'il n'existe aucun test totalement insatisfait, les diagnostics seront donnés par chacun des modes apparaissant dans les explications des tests douteux ($\mathbb{T}_{douteux} = \{test_i ; \mu(t_i) \in]0,1[\}$. Dans ce cas, l'indicateur *FORMAL* de chaque diagnostic $cfm\ (item)$ vaut $\mu(cfm\ (item)) = max_{test_i \in \mathbb{T}_{douteux}}(1 - \mu(test_i))$ (voir Eq. .0.12). Pour un diagnostic D_i en construction, deux cas se présentent :

– Il existe des tests totalement satisfaits. L'indicateur *FORMAL* est fixé à 100% si le diagnostic en construction contient des modes de défauts expliquant tous les tests totalement insatisfaits. Dans le cas contraire, l'indicateur *FORMAL* vaudra 0% : les anomalies sûres

Zones	(r,d)	μ_{test}
I	$r > r_{max}, d < d_{min}$	$\mu_{test} = 0$
II	$r_{min} < r < r_{max}, d < d_{min}$	$\mu_{test} = \frac{r_{max}-r}{r_{max}-r_{min}}$
III	$r < r_{min}, d < d_{min}$	$\mu_{test} = 1$
IV	$r > r_{max}, d_{min} < d < d_{max}$	$\mu_{test} = \frac{d-d_{min}}{2(d_{max}-d_{min})}$
V	$r_{min} < r < r_{max}, d_{min} < d < d_{max}$	$\mu_{test} = \frac{1+(2\frac{r_{max}-r}{r_{max}-r_{min}}-1)\frac{d_{max}-d}{d_{max}-d_{min}}}{2}$
VI	$r < r_{min}, d_{min} < d < d_{max}$	$\mu_{test} = \frac{1+\frac{d_{max}-d}{d_{max}-d_{min}}}{2}$
VII	$d > d_{max}$	$\mu_{test} = 0.5$

TAB. 5.1 – Calcul de μ_{test} dans les zones (r,d)

constatées ne sont pas encore expliquées d'un point de vue formel. Les cases *explication* dans la colonne (E) signifient, si elles sont cochées, que l'item de la ligne correspondante explique des tests totalement insatisfaits non encore expliqués.

– Il n'existe que des tests douteux. Dans ce cas, chaque mode de défaut présent dans un test douteux constitue à lui seul un diagnostic. Il existe donc un *ou logique* entre les modes de défauts du diagnostic en construction. Nous traduisons cela en logique floue par un opérateur de fuzzification qui ne sature pas : $max_{cfm\ (item)}\mu(cfm\ (item))$, où $\mu(cfm\ (item))$ est calculé comme rappelé précédemment. Les cases (E) n'ont pas de sens particulier dans ce cas : elles sont décochées.

L'indicateur *CONTEXTUAL*, noté aussi *C*, s'évalue quelque soit les résultats des tests et le diagnostic en construction. Nous avons montré que la distance dans $\mathbb{T}$ entre la signature effective et la signature associée à un diagnostic D_j donnée par l'équation 5.3.4. L'indicateur *CONTEXTUAL* est défini par : $contextual_{\mathbb{T}}(D_j) = 1 - distance_{\mathbb{T}}(D_j)$.

L'indicateur *A PRIORI*, noté aussi *A*, n'est pas affecté par le passage en logique floue.

Deux cas doivent être distingués pour la construction des suggestions des items apparaissant dans la colonne (D) de la matrice :

– Il existe des tests totalement satisfaits. L'ensemble des tests $\mathbb{T}^{insatisfait}$ est non vide. Il faut expliquer chacun de ces tests. Soient $\mathbb{T}^{insatisfait}_{expliqués}(D_j) \in \mathbb{T}^{insatisfait}$, l'ensemble des tests de $\mathbb{T}^{insatisfait}$ qui vérifient : $\{test_i \in \mathbb{T}^{insatisfait}; Expl(test_i) \cap D_j \neq \emptyset\}$, et son complémentaire $\mathbb{T}^{insatisfait}_{inexpliqués}(D_j)$ dans $\mathbb{T}^{insatisfait}$. D_j doit être complété afin de vider $\mathbb{T}^{insatisfait}_{inexpliqués}(D_j)$. Les items candidats affichés dans la colonne (D) sont donc donnés par $\bigcup_{test_i \in \mathbb{T}^{insatisfait}_{inexpliqués}(D_j)} Expl(test_i)$.
– Il n'existe que des tests douteux. Dans ce cas, seuls les modes des tests douteux plutôt insatisfaits sont présentés ($\mu(test) < 50\%$) à l'exception des modes qui appartiennent déjà à D_j : $\{\bigcup_{\{test_i} \in \mathbb{T}; \mu(test_i) < 50\%\} Expl(test_i) \backslash D_j\}$. Si la case *all* est cochée, les modes de tous les tests sont présentés : $\{\bigcup_{\{test_i} \in \mathbb{T}\} Expl(test_i) \backslash D_j\}$.

Reste encore a étudier comment la rétro-analyse peut être adaptée au contexte flou. Nous

avons vu dans la partie 5.2.3, que la rétro-analyse consistait à analyser les performances des tests sachant que l'état réel, noté D^*, du système à diagnostiquer est connu.

Nous avons aussi vu que le degré de satisfaction d'un test s'écrivait (voir équation .0.9) :

$$\mu(test) = \frac{1 + (2\mu_{\mathscr{K}} - 1)\mu_{\mathscr{K}'}}{2}$$

L'algorithme proposé dans la partie 5.2.3 devient alors :

D^* : **un ensemble de modes correspondant à l'état réel du système**
tests : **un ensemble de tests effectués sur le système**
Pour $test \in tests$ **faire**
  Si $(Expl(test) \cap D^* = \emptyset)$ **Alors**
    [le degré de fausse alarme vaut $1 - \mu(test)$ *(1=critique)]*
  Fin Si
  Si $(Expl(test) \cap D^* \neq \emptyset)$ **Alors**
    [le degré de non-détection vaut $\mu(test)$*]*
  Fin Si
Fin Pour

5.3.3 Estimation des résultats

Dans cette partie, nous allons présenter un scénario d'interaction homme-automate d'analyse diagnostique avec intégration du raisonnement diagnostique flou. Ensuite, nous allons comparer les résultats donnés sur les huit mois du problème Hydrodiag dans le cas de la logique nette et celui de la logique floue pour examiner ce qu'apporte la logique floue.

Exemple de scénario de diagnostic

Revenons à l'exemple présenté dans la rubrique 5.2.4. Pour comparer les résultats donnés par le seuil de détection net et le seuil flou, nous prenons le même scénario correspondant à la recherche des pluviomètres défaillants durant le mois d'août 2002. A partir des données collectées des pluviomètres et le retour de l'expert sur les défauts, les paramètres de seuil de détection floue sont déterminés par l'optimisation sous contraintes tel que défini par les propositions 5.3.7 et 5.3.8 page 149. La solution est donnée sur la figure 5.17 avec $r_{max} = 28.2\ mm/m^2/h$, $r_{min} = 0\ mm/m^2/h$, $d_{min} = 3.6703\ (Km)$, $d_{max} = 10\ (Km)$.

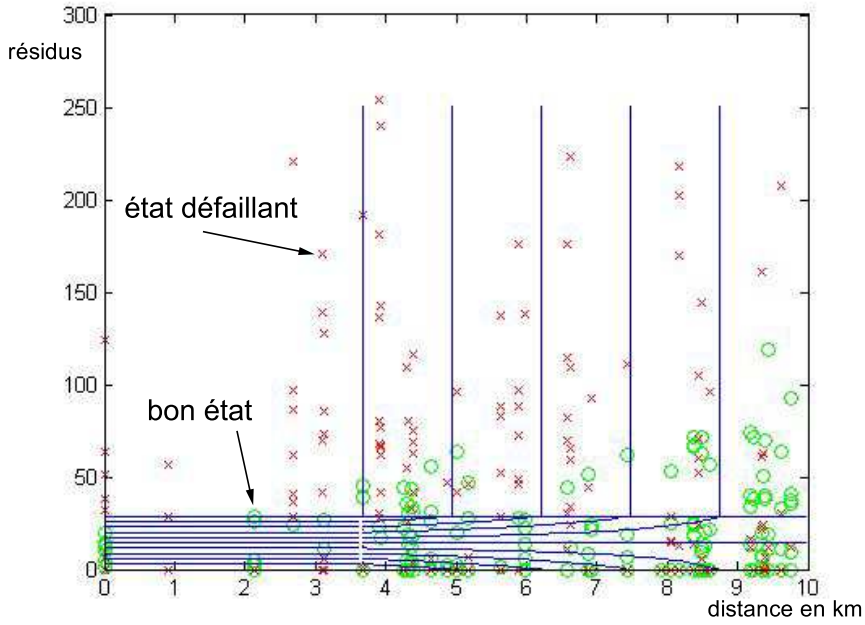


FIG. 5.17 – Seuil de détection dans le cas flou

Un ensemble de 159 diagnostics possibles sont trouvés par le logiciel. L'interaction homme-automate commence par la détermination du diagnostic le plus probable sachant que 159 diagnostics ne sont pas exploitables sous forme de liste.

```
------------------------------
diagnosis #0 (F:100.0%, C:85.67053389481522%, A:1.0000000000000005E-8%)
Component d625 (zoumboubani) is faulty
 and component d626 (dapefougou) is faulty
 and component d630 (akekerou) is faulty
 and component d641 (banikani) is faulty
 and component d611 (donga) is faulty
 and component d644 (gountia) is faulty
 and component d645 (koko-sika) is faulty
 and component d614 (adiangdia) is faulty
 and component d647 (parakou_2) is faulty
 and component d639 (kolokonde) is faulty
------------------------------
..............................
------------------------------
diagnosis #159 (F:100.0%, C:77.07930039614439%, A:1.0000000000000006E-10%)
Component d627 (ananinga) is faulty
 and component d629 (gaouga) is faulty
```

```
 and component d625 (zoumboubani) is faulty
 and component d628 (bombone) is faulty
 and component d651 (nalohou_2) is faulty
 and component d611 (donga) is faulty
 and component d642 (barienou) is faulty
 and component d643 (gangamou) is faulty
 and component d645 (koko-sika) is faulty
 and component d632 (adiangdia_oues) is faulty
 and component d636 (parakou) is faulty
 and component d639 (kolokonde) is faulty
------------------------------
```

L'expert découvre la matrice de diagnostic sur la figure 5.18.

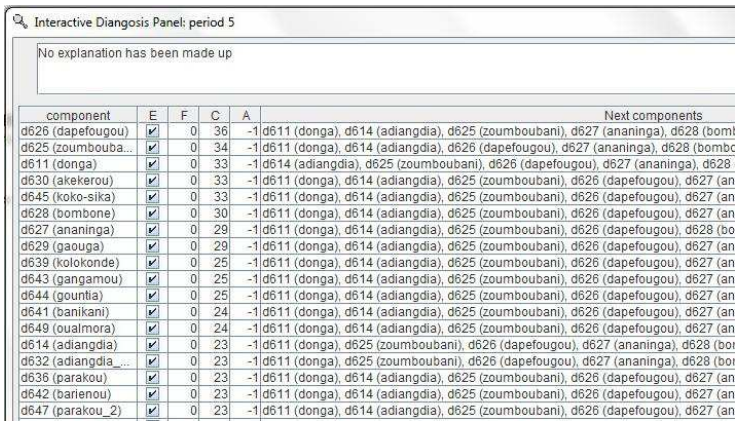
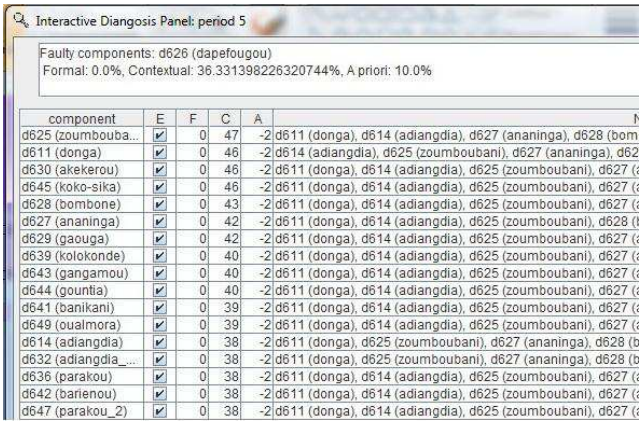


FIG. 5.18 – Analyse diagnostique pour le mois d'août 2002 - Etape 1

Comme l'indicateur *CONTEXTUAL* est plus important pour d626 (d626 explique beaucoup de symptômes), l'expert choisit naturellement ce diagnostic (figure 5.19). La position du capteur s'allume alors en bleu sur la carte du logiciel *hydrodiag* et une fenêtre avec les hyétogrammes des capteurs situés dans un rayon de 10km apparaît (voir figure 5.20). Visiblement, le capteur d626 est défaillant. L'expert poursuit cette direction explicative.

Comme l'indicateur *CONTEXTUAL* est plus important pour d625, l'expert choisit ce diagnostic (figure 5.21). L'expert observe ensuite les hyétogrammes des capteurs situés dans un rayon de 10km autour de d625 (figure 5.22). Par sa connaissance implicite, l'expert indique que le capteur d625 est défaillant. Il poursuit cette direction explicative.

De la même manière, l'expert continue la procédure de diagnostic jusqu'à ce qui'il puisse établir un diagnostic pouvant expliquer tous les symptômes. Jusqu'ici, aucun élément de la colonne (E) du matrice de diagnostic n'est coché (figure 5.23). Les pluviomètres défaillants sont donc : d626 (dapefougou), d625 (zoumboubani), d645 (koko sika), d611 (donga), d643



component	E	F	C	A	
d625 (zoumbouba...	✔	0	47	-2	d611 (donga), d614 (adiangdia), d627 (ananinga), d628 (bom
d611 (donga)	✔	0	46	-2	d614 (adiangdia), d625 (zoumboubani), d627 (ananinga), d62
d630 (akekerou)	✔	0	46	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d645 (koko-sika)	✔	0	46	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d628 (bombone)	✔	0	43	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d627 (ananinga)	✔	0	42	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d628 (
d629 (gaouga)	✔	0	42	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d639 (kolokonde)	✔	0	40	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d643 (gangamou)	✔	0	40	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d644 (gountia)	✔	0	40	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d641 (banikani)	✔	0	39	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d649 (oualmora)	✔	0	39	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d614 (adiangdia)	✔	0	38	-2	d611 (donga), d625 (zoumboubani), d627 (ananinga), d628 (b
d632 (adiangdia_...	✔	0	38	-2	d611 (donga), d625 (zoumboubani), d627 (ananinga), d628 (b
d636 (parakou)	✔	0	38	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d642 (barienou)	✔	0	38	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (
d647 (parakou_2)	✔	0	38	-2	d611 (donga), d614 (adiangdia), d625 (zoumboubani), d627 (

FIG. 5.19 – Analyse diagnostique pour le mois d'août 2002 - Etape 2

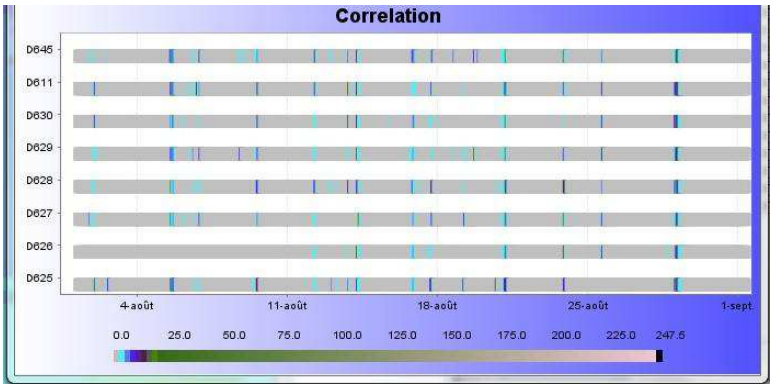


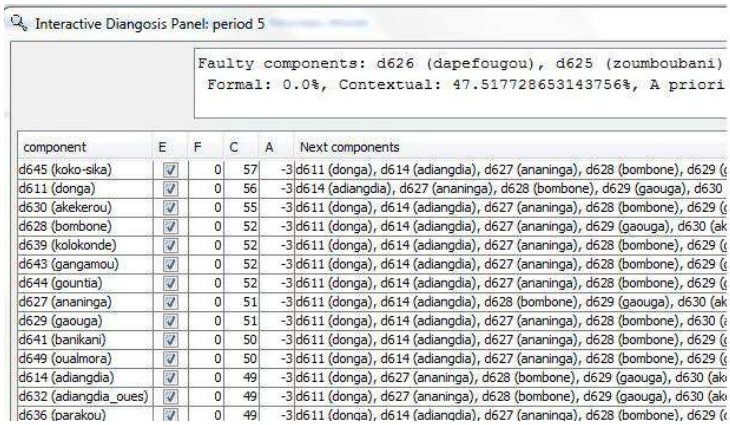
FIG. 5.20 – Analyse diagnostique pour le mois d'août 2002 - Etape 2

(gangamou), d630 (akekerou), d641 (banikani), d649 (oualmora), d632 (adiangdia oues), d636 (parakou), d642 (barienou).

Comparaison entre les résultats donnés par le seuil de détection net et le seuil de détection flou

Nous évaluons maintenant l'intérêt des *seuils de détection flous* en comparaison avec les *seuils de détection nets* à l'aide de l'ensemble des diagnostics que nous pouvons obtenir sur les 8 périodes. Chaque période correspond à un mois de mars à octobre 2002. Les diagnostics sont représentés sur le tableau de la figure 5.24.

Rappelons les paramètres du seuil de détection trouvé : $r_{max} = 28.2\ mm/m^2/h$, $r_{min} = 0$, $d_{min} = 3.6703\ (km)$, $d_{max} = 10\ (km)$. Sur la figure 5.17, nous voyons que les paramètres $r_{max} = 28.2$ et $d_{min} = 3.6703\ (km)$ permet de garantir qu'il n'y pas de fausse-alarme. $r_{min} = 0\ mm/m^2/h$ permet d'éviter toutes les non-détections. Dans ce cas, la zone de doute est large (μ_{test} a une valeur entre 0 et 1). Ceci explique pourquoi avec le seuil de détection flou, nous



component	E	F	C	A	Next components
d645 (koko-sika)	☑	0	57	-3	d611 (donga), d614 (adiangdia), d627 (ananinga), d628 (bombone), d629 (g
d611 (donga)	☑	0	56	-3	d614 (adiangdia), d627 (ananinga), d628 (bombone), d629 (gaouga), d630
d630 (akekerou)	☑	0	55	-3	d611 (donga), d614 (adiangdia), d627 (ananinga), d628 (bombone), d629 (g
d628 (bombone)	☑	0	52	-3	d611 (donga), d614 (adiangdia), d627 (ananinga), d629 (gaouga), d630 (ak
d639 (kolokonde)	☑	0	52	-3	d611 (donga), d614 (adiangdia), d627 (ananinga), d628 (bombone), d629 (g
d643 (gangamou)	☑	0	52	-3	d611 (donga), d614 (adiangdia), d627 (ananinga), d628 (bombone), d629 (g
d644 (gountia)	☑	0	52	-3	d611 (donga), d614 (adiangdia), d627 (ananinga), d628 (bombone), d629 (g
d627 (ananinga)	☑	0	51	-3	d611 (donga), d614 (adiangdia), d628 (bombone), d629 (gaouga), d630 (ak
d629 (gaouga)	☑	0	51	-3	d611 (donga), d614 (adiangdia), d627 (ananinga), d628 (bombone), d630 (a
d641 (banikani)	☑	0	50	-3	d611 (donga), d614 (adiangdia), d627 (ananinga), d628 (bombone), d629 (g
d649 (oualmora)	☑	0	50	-3	d611 (donga), d614 (adiangdia), d627 (ananinga), d628 (bombone), d629 (g
d614 (adiangdia)	☑	0	49	-3	d611 (donga), d627 (ananinga), d628 (bombone), d629 (gaouga), d630 (ak
d632 (adiangdia_oues)	☑	0	49	-3	d611 (donga), d627 (ananinga), d628 (bombone), d629 (gaouga), d630 (ak
d636 (parakou)	☑	0	49	-3	d611 (donga), d614 (adiangdia), d627 (ananinga), d628 (bombone), d629 (g

FIG. 5.21 – Analyse diagnostique pour le mois d'août 2002 - Etape 2

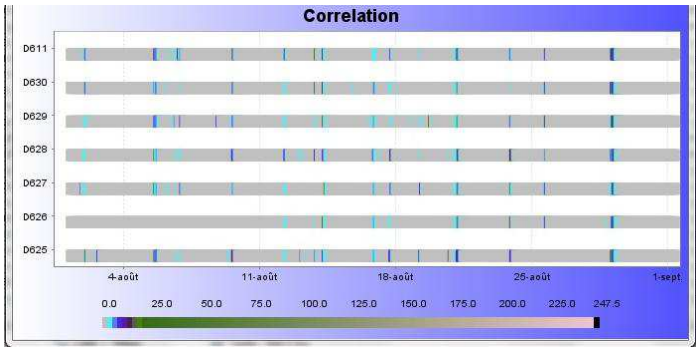


FIG. 5.22 – Analyse diagnostique pour le mois d'août 2002 - Etape 2

avons plus de diagnostics qu'avec un seuil de détection net. Chaque diagnostic est constitue de plusieurs capteurs en défaut (voir le tableau sur la figure 5.24).

Nous voyons que le seuil de détection flou permet de mieux détecter les pluviomètres en défaut en réduisant le nombre de non-détections. La finesse des résultats requiert néanmoins une finesse d'anlyse de l'expert. Comme la détection s'appuie sur la corélation de paires de pluviomètres, si une paire est détectée en défaut, l'hydrologue va analyser les hyétogrammes d'autres pluviomètres situés autour des pluviomètres dans cette paire (dans un rayon de 10km) pour déterminer lequel est en panne. Pour les détections sensibles, l'observation sur l'hyétogramme demande beaucoup d'expériences de l'hydrologue. Cette détection est parfois fausse à cause de l'écart de quantité de pluie entre deux zones éloignées. Prenons l'exemple d'une détection sensible pour le diagnostic obtenue durant la période 3 représenté sur la figure 5.25. Sur cette figure {d626, d627, d614, d636} est un diagnostic en construction. Les mesures contextuelles de d629, d625 et d645 sont les plus importantes. Supposons que l'expert choisisse d629, une fenêtre avec les hyétogrammes des capteurs situés dans un rayon de 10km de d629 apparaît. Sur ces hyétogrammes, il est très difficile de déterminer quel capteur parmi d627, d625 et d645 est en défaut.

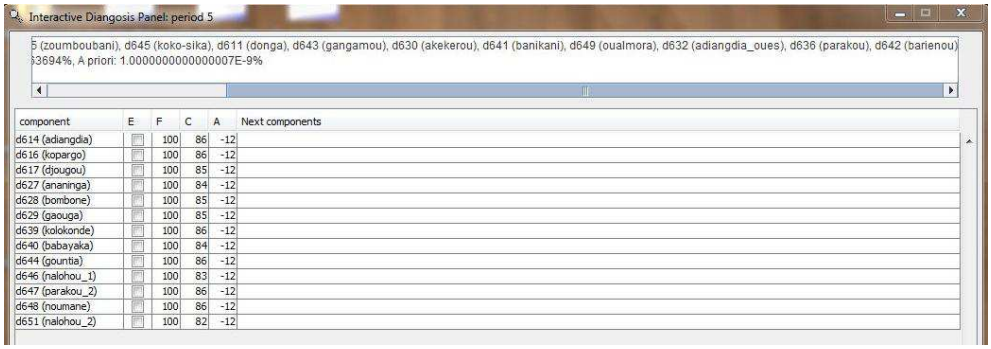


FIG. 5.23 – Analyse diagnostique pour le mois d'août 2002 - Etape 2

5.4 Conclusion

Dans ce chapitre, nous avons présenté un processus de diagnostic avec interaction homme-automate dans la phase de diagnostic pour exploiter l'expertise implicite. Le cas d'étude présenté est un problème de détection de défaut sur un réseau de pluviomètres (Nguyen [2005]). En plus des travaux présentés dans (Nguyen [2005]), cet exemple montre comment les connaissances tacites des hydrologues peut être exploitées durant la phase de diagnostic pour affiner le résultat de diagnostic à partir d'un ensemble important de propositions. Avec une connaissance non-explicitée complémentaire, un expert peut déterminer si un capteur est défaillant ou pas en regardant son hyétogramme et ceux de ses voisins alors que le système d'aide ne peut le détecter car les tests ne peuvent pas intégrer toute la connaissance experte. Un outil d'aide au diagnostic a été conçu pour accompagner l'hydrologue dans sa recherche des capteurs en défaut.

A côté du seuil de détection net (inconsistant/consistant), les résultats de la logique floue dans l'analyse diagnostique (Touaf et Ploix [2004b]) ont été appliqués dans ce travail. Des seuils de détection flous ont donc été conçus autant pour le test de détection que pour le test de validité. En comparaison avec le seuil net, un seuil flou permet de réduire le nombre de non-détections et conduit à des dignostics plus précis au prix d'une plus grande expertise.

Périodes	Seuil de détection net		Seuil de détection flou		Commentaires
	Nombre de diagnostics possibles	Diagnostics trouvés	Nombre de diagnostics possibles	Diagnostics trouvés	
0	0	Système est ok	4	d626, d627, d628, d639	Il y a plus de diagnostics avec le seuil de détection flou qu'avec le seuil de détection net. Il y a plusieurs détections sensibles dans le cas du seuil de détection flou. Il nécessite donc beaucoup d'expériences de l'expert pour évaluer les hyétogrammes (quantité de pluie).
1	16	d614, d636, d626, d627	32	d626, d627, d616, d639, d614, d636	
2	8	d614, d626, d627, d636	384	d626, d627, d614, d636, d629, d645, d625, d616, d611, d617, d641, d646, d639	
3	4	d614, d627	72	d627, d626, d640, d643, d648, d614, d650, d617, d642	
4	12	d626, d639, d648, d643, d625	152	d626, d639, d648, d643, d625, d611, d640, d629, d630, d651, d614, d642	
5	16	d643, d632, d647, d626	160	d626, d625, d645, d611, d643, d630, d641, d649, d632, d647, d642	
6	80	d626, d643, d645, d647, d614, d617, d611, d628	360	d626, d629, d645, d611, d627, d646, d628, d643, d616, d614, d617, d644, d647	
7	32	d614, d647, d643, d626, d617	432	d626, d628, d629, d651, d617, d630, d645, d640, d643, d614, d642, d647	

FIG. 5.24 – Résultats de diagnostic donnés par le seuils de détection net et le seuil de détection flou

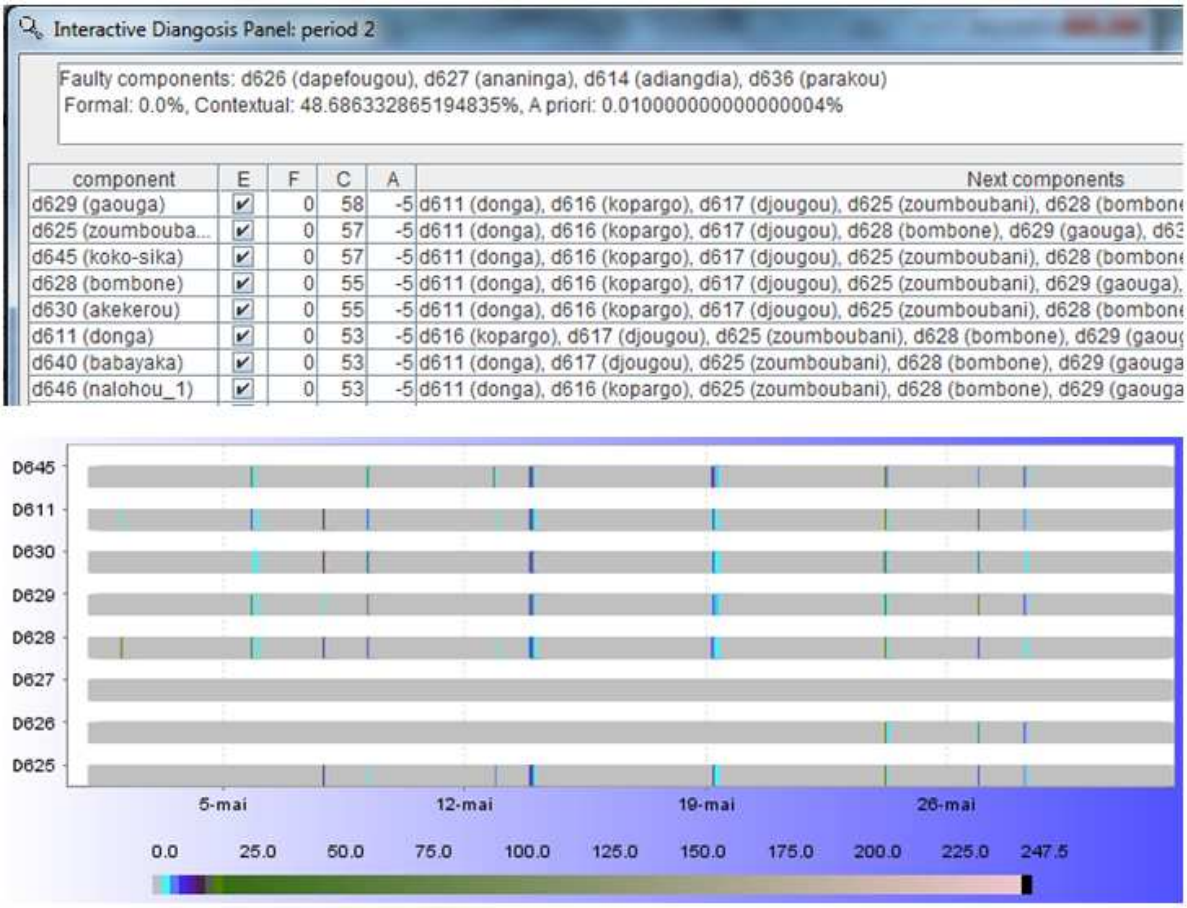


FIG. 5.25 – Exemple d'une détection sensible avec le seuil de détection flou

Chapitre 6

Conclusions et perspectives

Les travaux que nous avons présentés dans ce manuscrit consistent à étudier les différentes formes que peut prendre un problème de diagnostic dans un contexte itératif. Le premier objectif est de proposer des outils pour aider des agents de maintenance dans leur travail de dépannage. Les systèmes à dépanner sont variés et parfois complexes ; les données recueillies, mais aussi les connaissances qui peuvent être raisonnablement explicitées, peuvent être de natures très différentes. Dans ce contexte, l'analyse diagnostique telle qu'elle est généralement posée dans la littérature scientifique n'est pas toujours applicable parce qu'il peut être trop long et trop coûteux de formaliser toute la connaissance d'un expert sur un système à diagnostiquer dans un modèle complet construit a priori. Il n'est pas toujours nécessaire de modéliser des parties qui n'apportent rien à un raisonnement diagnostique sachant qu'une situation de défaut dans un système complexe et souvent unique. Le deuxième objectif est d'exploiter la connaissance implicite de l'expert qui ne peut pas être formulée sous forme du modèle formel. La démarche de l'analyse diagnostique est donc présentée comme un processus avec interactions homme-automate. Dans cette démarche, les informations sont exploitées au fur et à mesure pour localiser les défauts.

Notre travail a consisté à analyser les différentes situations où un problème de diagnostic où des intéractions homme-automate sont nécessaire. Nous avons recensé trois types de difficultés qui se rapportent toutes à la complexité du modèle du système :

- difficulté liée au grand nombre d'éléments d'un système
- difficulté liée à la représentation du comportement / fonctionnement d'un système
- difficulté liée à l'explicitation de l'expertise

En ce qui concerne la complexité d'un système, le problème de diagnostic est présenté comme un processus itératif avec interactions homme-automate, où le modèle du système est découvert et explicité au fur et à mesure de l'avancement du processus. L'abstraction entre les éléments de modélisation est donc abordée.

Dans un premier temps, notre contribution réside dans la formulation d'un modèle d'abstraction. En comparaison avec les travaux existants dans la littérature, cette formulation s'appuie sur un point de vue orienté-objet. Le modèle hiérarchique est défini via les éléments de base d'un système : variable, modèle élémentaire et item (fonction ou ressource). Ce sont les éléments que l'expert ajoute au fur et à mesure de la construction du modèle.

Les contributions des chapitres 3 et 4 ont conduit à la proposition d'une démarche de diagnostic centré autour d'un modèle hiérarchique construit à chaque itération d'un processus de diagnostic. Dans le chapitre 3, la modélisation structuro-fonctionnelle FIS permet de décrire un système à différents niveaux de détails. Les relations hiérarchiques entre les items sont gérées dans le calcul des diagnostics telles qu'elles ne perturbent pas les résultats. Dans le chapitre 4, les modèles comportementaux sont intégrés dans le modèle FIS. En comparaison avec le problème proposé dans le chapitre 3, les symptômes sont détectés automatiquement dans le chapitre 4. Pour collecter des symptômes, nous avons appliqué une méthode structurelle pour rechercher les sous-systèmes testables en gérant les relations hiérarchiques entre les items telles qu'elles ne perturbent pas les résultats de diagnostics. La méthode de propagation de valeurs inspirée du problème de satisfaction de contraintes (CSP) est appliquée pour chaque SST afin de vérifier la consistance entre chaque SST et les observations fournies par l'expert.

Concernant l'expertise non-formulée, un processus avec interactions durant la phase de diagnostic est présenté dans le chapitre 5. Le cas d'étude présenté est le problème de détection de défaut sur un réseau de pluviomètres distribués sur un bassin versant. Ce cas d'étude permet de montrer comment la connaissance non-formulée d'un hydrologue peut être exploitée durant la phase de diagnostic. En comparaison avec les travaux existants, notre contribution se focalise tout d'abord sur la conception d'une matrice interactive de diagnostic pour exploiter la connaissance implicite de l'expert d'une part et pour guider l'expert à établir un diagnostic à partir d'un ensemble de propositions d'autre part. Pour déterminer des seuils de détection permettant de représenter les doutes que peut avoir un expert sur le résultat d'un test, nous avons montré que la logique floue pouvait être avantageusement utilisée. Ceci permet d'optimiser les non-détections dans la phase de test.

Les approches itératives et interactives proposées constituent des extensions aux travaux de la littérature scientifique sur l'analyse diagnostique car elles correspondent au cas où il n'existe qu'une itération.

Les travaux de ce mémoire ont étudiés différentes formes de processus de diagnostic et ont conduit à proposer différents outils d'aide. Il est évident que seule une partie des processus où des interactions homme-automate peuvent constituer une aide ont été étudiées. Des approches d'aide interactive au recueil d'observations avec une aide à l'expert sur le type d'observation qu'il serait préférable d'effectuer présenteraient un intérêt certain. Ces approches conduiraient à coupler les outils que nous avons proposé avec des outils de placement de capteurs. Pour compléter ces travaux, d'autres perspectives peuvent être envisagées :

– traiter le cas où l'expert définit des modes de défaut spécifiques pour chaque item et les

liens entre les modes des items (OU/ET). Il peut donc exister à la fois des OUs et des ETs logiques dans $Expl(\top(T_i))$ comme nous l'avons expliqué dans la remarque 1, page 75.

– traiter des modèles de comportement dont le domaine des variables est réel. Ceci concerne la partie de propagation de valeur au sein de chaque SST pour collecter des symptômes. Dans notre travail, nous avons appliqué la méthode de propagation de valeurs mais cette méthode ne s'applique qu'à des variables dont le domaine de valeurs est discret. Des approches par intervalles permettraient certainement de résoudre ces problèmes.
– propager directement sur les contraintes décrivant le système (ne pas utiliser des SST) pour collecter des symptômes afin de pouvoir suggérer les valeurs à mesurer à l'expert.
– développer et améliorer les outils logiciels présentés dans les chapitres 3 et 4.

Annexe A

Algorithmes de diagnostic logique

Algorithme de Reiter (Reiter [1987])

Dans la littérature, l'algorithme de HS-Tree de Reiter (Reiter [1987]) est exploité pour calculer des diagnostics à partir des explications des symptômes. Cet algorithme permet de prendre en compte des défauts multiples. Nous résumons tout d'abord l'idée principale de l'algorithme de HS-Tree, puis, nous allons montrer comment l'ensemble des modes impossibles sera intégré.

Approche de diagnostic de Reiter

Dans (Reiter [1987]), un problème de diagnostic est défini par $(SD, COMPONENTS, OBS)$ avec SD la description du système, $COMPONENTS$ est un ensemble des composants, OBS est un ensemble d'observations. Reiter propose une approche qui s'appuie sur la conception d'un conflit (conflict set). *Un conflit* pour $(SD, COMPONENTS, OBS)$ est un ensemble $\{c_1, \ldots, c_n\} \subseteq COMPONENTS$ tel que $SD \cup OBS \cup \{\neg AB(c_1), \ldots, \neg AB(c_n)\}$ est inconsistant. Un conflit S_i pour $(SD, COMPONENTS, OBS)$ est minimal s'il n'existe pas un sous ensemble de S_i qui est aussi un conflit pour $(SD, COMPONENTS, OBS)$. En faisant le lien avec la notion de *test de détection* proposé dans (Ploix *et al.* [2003]), un *conflit* correspond à l'explication d'un test inconsistant.

Un *hitting set* d'un ensemble de conflits $C = \{S_1, \ldots, S_n\}$ est un ensemble $H \subseteqq \bigcup_{S \in C} S$ tel que $H \cap S \neq \emptyset$. H est minimal s'il n'existe pas un sous-ensemble de H qui est aussi un *hitting set* pour C.

En s'appuyant sur les notions de *conflit* et de *hitting set*, les deux résultats principaux présentés dans (Reiter [1987]) sont : (1) le théorème sur la notion de diagnostic et (2) une approche pour calculer les diagnostics à partir d'un ensemble des conflits. Cette approche est appelé l'arbre HS (Hitting-Set Tree).

Définition .0.1. $\Delta \subseteq COMPONENTS$ *est un diagnostic pour* $(SD,\ COMPONENTS,\ OBS)$

si et seulement si Δ *est un hitting set minimal pour un ensemble de conflits de* $(SD, COMPONENTS, OBS)$.

L'ensemble des *hitting sets* est construit par l'arbre HS. Soit C l'ensemble des conflits et S un élément de C. L'algorithme de l'arbre HS permet de chercher tous les hitting sets en gardant l'arbre HS aussi petit que possible (Reiter [1987]), corrigé par (Greiner *et al.* [1989]).

1. *Étiqueter un noeud n*
   Supposons que H(n) est un hitting set obtenu à un noeud n sur l'arbre HS. H(n) est représenté par l'ensemble des éléments étiquetés sur les branches de la chemin du noeud n à la racine. Si $\exists S \in C$ tel que $H(n) \cap S = \emptyset$, le noeud n est étiqueté par S, si non n est étiqueté $\checkmark$.
2. *Étiqueter une branche*
   Si un conflit S est étiqueté à un noeud n, les branches descendant sont étiquetées par les éléments impliqués dans S.
3. *Élaguer des branches*
   Si un noeud n est déjà généré et s'il y a un autre noeud n' tel que $H(n) = H(n')$, on ferme n'. Un noeud formé est étiqueté par $\times$.
   Si un noeud n est étiqueté par $\checkmark$, et il y a un autre noeud n' tel que $H(n) \subseteq H(n')$, on ferme le noeud n' par $\times$.
   Si un noeud n est étiqueté par $\checkmark$, et il y a un autre noeud n' qui est étiqueté par $\checkmark$tel que $H(n') \subset H(n)$, alors le noeud n est fermé. n est réétiqueté par $\times$.

L'analyse diagnostique commence par des symptômes collectés par l'expert. Il existe implicitement un *ET* logique entre les symptômes. Les modes des items parent impliqués dans les explications des symptômes seront remplacés par des équivalences logiques obtenues.

Exemple 18. Pour illustrer l'utilisation de l'algorithme de Reiter, revenons à l'exemple des deux cuves (exemple 2, page 27). Supposons que trois tests négatifs sont obtenus dont les explications sont données par :

$$\top(T_1) \rightarrow \neg ok(bac1) \vee \neg ok(capteur1) \vee \neg ok(vanne1)$$
$$\top(T_2) \rightarrow \neg ok(bac1) \vee \neg ok(bac2) \vee \neg ok(capteur2) \vee \neg ok(vanne1) \vee \neg ok(vanne2)$$
$$\top(T_3) \rightarrow \neg ok(bac1) \vee \neg ok(bac2) \vee \neg ok(capteur1) \vee \neg ok(capteur2) \vee \neg ok(vanne2)$$

Le support de chaque explication correspond à un conflit selon la définition de Reiter.

$$S_1 : (\neg ok(bac1), \neg ok(capteur1), \neg ok(vanne1))$$

$$S_2 : (\neg ok(bac1), \neg ok(bac2), \neg ok(capteur2), \neg ok(vanne1), \neg ok(vanne2))$$
$$S_3 : (\neg ok(bac1), \neg ok(bac2), \neg ok(capteur1), \neg ok(capteur2), \neg ok(vanne2))$$

Un arbre HS peut être construit à partir de $C = \{S_1, S_2, S_3\}$ (figure 1)

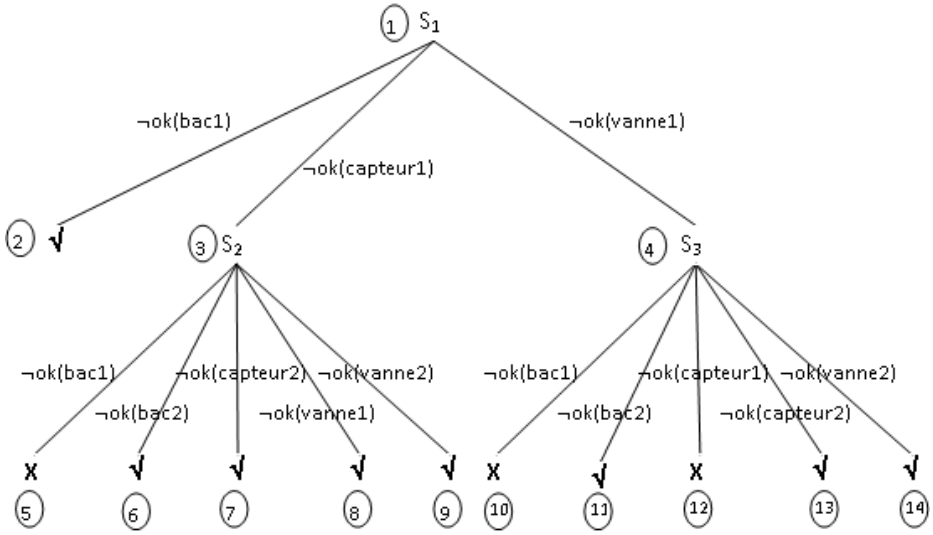


FIG. 1 – Génération des *hitting set* par HS-Tree

Les noeuds sont générés en largeur de haut en bas selon un ordre marqué sur la figure (1). Le noeud (5) est fermé par ce qu'il existe déjà sur cet arbre le noeud (2) avec $H(2) \subset H(5)$, on proceède ainsi pour tout l'arbre. Les diagnostics sont déduits :

$$\begin{aligned}
&(\neg ok(bac1)) \\
&(\neg ok(capteur1) \wedge \neg ok(bac2)) \\
&(\neg ok(capteur1) \wedge \neg ok(capteur2)) \\
&(\neg ok(capteur1) \wedge \neg ok(vanne1)) \\
&(\neg ok(capteur1) \wedge \neg ok(vanne2)) \\
&(\neg ok(vanne1) \wedge \neg ok(bac2)) \\
&(\neg ok(vanne1) \wedge \neg ok(capteur2)) \\
&(\neg ok(vanne1) \wedge \neg ok(vanne2))
\end{aligned} \tag{.0.1}$$

Gestion des modes multiples

Lorsque le modèle de défaut multiple est considéré, l'algorithme standard de HS-Tree peut conduire à des diagnostics qui contiennent plusieurs modes de comportements d'un même

item. Ces diagnostics sont impossibles par ce qu'un item n'est qu'en un seul mode à un moment donné. Cette problème a été abordé aussi dans De Kleer et Williams [1992]. Le simple méthode proposé par De Kleer est d'enlever les diagnostics qui comportent à la fois plusieurs modes d'un même item en même temps. Pourtant, il n'a pas montré comment intégrer cette principe dans l'algorithme de diagnostic. Nous allons proposer dans cette partie une méthode pour éviter la co-existance de plusieurs modes d'un même item dans un diagnostic en s'appuisant sur l'algorithme standard de HS-Tree.

Pour éviter la co-existance de plusieurs modes d'un même item dans un diagnostic, la notion de *chemin couvrant* est introduite. Soient S_n un ensemble de modes d'un conflit étiqueté à un noeud, $\{mode_1(I_1), mode_2(I_2), \ldots, mode_n(I_n)\}$ est l'ensemble de modes étiquetés sur les branches appatenant au chemin de S_n à la racine, notons par $path(S_n)$. Le *couvrant* de $path(S_n)$ est donné par $covering(path(S_n)) = \{Modes(I_1) \cap \cdots \cap Modes(I_n)\}$. Ainsi, plusieurs modes peuvent être gérés de la manière suivante : dans le développement d'un arbre HS, chaque branche descendant d'un noeud S_n est étiquetée par un mode impliqué dans S_n. Pour éviter la coexistence des modes différents d'un même item dans un diagnostic, quand on fragmente un noeud S, un mode $mode(I) \in S_n$ conduit à une nouvelle branche si $mode(I) \notin covering(path(S_n))$.

Classement des diagnostics

Mesure de coïncidence des défauts

Afin de garder un raisonnement rigoureux, les tests positifs ne sont pas pris en compte pour le calcul des diagnostics, à l'exception des *tests positif complètement testés*. Les résultats des tests positifs normaux sont utiles pour le classement des diagnostics. Dans (Ploix *et al.* [2003]), une approche basée sur la distance entre la signature théorique et la signature effective des tests a été proposée.

La signature des modes est définie via une table de signature. La table de signature ne comporte pas directement des contraintes dans les colonnes mais des modes. Il y a une différence car plusieurs contraintes différentes peuvent se rapporter à un même mode.

Définition .0.2. *(table de signature) Une table de signatures est une représentation matricielle $\mathscr{T} : Modes(T) \rightarrow T$, des modes associées à un ensemble de tests T. Soit $T = [t_i]$, un ensemble de tests et $Modes = [m_j]$, l'ensemble des modes intervenant dans les descriptions ayant servies à construire T. La table de signature T est une matrice définie par :*

$$\mathscr{T} = [\mathscr{T}_{i,j}] / \begin{cases} \mathscr{T}_{i,j} = 1 \text{ si } m_j \in modes(t_i) \\ \mathscr{T}_{i,j} = 0 \text{ si } m_j \notin modes(t_i) \end{cases}$$

où $modes(t_i)$ désigne les modes des descriptions ayant servies à construire t_i.

La signature d'un mode est donnée par une table de signature.

Définition .0.3. *(signature) La signature d'un mode mode* $\in Modes(T)$ *dans la table de signature* $\mathscr{T} : Modes(T) \rightarrow T$*, associé à un ensemble de tests, est égale au vecteur correspondant à la colonne liée à mode.*

Dans le cas des défauts multiples, un diagnostic est présenté sous forme d'une conjonction de modes. La signature théorique d'une conjonction de mode est obtenue par l'application de l'opérateur logique *OU* entre les vecteurs qui représentent les signatures des modes impliqués dans la conjonction. Par exemple nous avons la signature théorique de deux modes $\sigma_T(mode(I_i)) = (1\ 0\ 1)$ et $\sigma_T(mode(I_j)) = (0\ 1\ 1)$, nous avons $\sigma_T(mode(I_i) \wedge mode(I_j)) = (0\ 0\ 1)$.

Signature effective Soit $T = (t_i)$ est une liste ordonné des tests. A un instant donné, la signature effective dans T, noté par σ_T^*, est donnée par :

$$\forall i, \begin{cases} (\sigma_T^*)_i = 1 \leftrightarrow t_i \text{ est inconsistant} \\ (\sigma_T^*)_i = 0 \leftrightarrow t_i \text{ est consistant} \end{cases}$$

La mesure de coïncidence consiste à mesurer la distance de Hamming entre la signature théorique et la signature effective d'un diagnostic (Ploix *et al.* [2003]). Soit $T = (t_i)$ une liste ordonnée des tests, et $D = d_i$ un ensemble de diagnostics. La mesure de coïncidence est donnée par :

$$\forall d_i \in D, \mu_T^c(d_i) = \frac{|\sigma_T(d_i), \sigma_T^*|_{Hamming}}{card(T)} \tag{.0.2}$$

La haute valeur de la mesure de coïncidence d'un mode de défaut montre que ce mode est moins probable.

Exemple 19. Revenons à l'exemple de deux cuves (exemple 18) (aussi détaillé dans l'exemple 2, page 27). La table de signature 1 est obtenu.

	$ok(bac1)$	$ok(bac2)$	$ok(capteur1)$	$ok(capteur2)$	$ok(vanne1)$	$ok(vanne2)$
t_1	1	0	1	0	1	0
t_2	1	1	0	1	1	1
t_3	1	1	1	1	0	1

TAB. 1 – Table de signatures du système de cuves

Pour illustrer l'intérêt de la mesure de coïncidence, prenons maintenant un autre scénario où les tests T_1 et T_2 sont inconsistants et le test T_3 consistant. Le calcul de diagnostic par l'arbre-HS à partir de $\top(T_1)$ et $\top(T_2)$ nous donne les diagnstics :

$$(\neg ok(bac1))$$
$$(\neg ok(vanne1))$$
$$(\neg ok(capteur1) \wedge \neg ok(bac2))$$
$$(\neg ok(capteur1) \wedge \neg ok(capteur2))$$
$$(\neg ok(capteur1) \wedge \neg ok(vanne2))$$

Notons que quand les modes de défauts spécifiques des items ne sont pas définis, $\neg ok(I) \equiv cfm\ (I)$. La signature effective des tests est $\sigma_T^* = (1\ \ 1\ \ 0)$. Les signatures théoriques des diagnostics et les mesures de coïncidence des défauts sont donnés par le tableau (2).

Signature théorique	Mesure de coïncidence
$\sigma_T(cfm\ (bac1)) = (1\ \ 1\ \ 1)$	$\mu_T^c(cfm\ (bac1)) = 33.33$
$\sigma_T(cfm\ (vanne1)) = (1\ \ 1\ \ 0)$	$\mu_T^c(cfm\ (vanne1)) = 0.00$
$\sigma_T(cfm\ (capteur1) \wedge cfm\ (bac2)) = (1\ \ 1\ \ 1)$	$\mu_T^c(cfm\ (capteur1) \wedge cfm\ (bac2)) = 33.33$
$\sigma_T(cfm\ (capteur1) \wedge cfm\ (capteur2)) = (1\ \ 1\ \ 1)$	$\mu_T^c(cfm\ (capteur1) \wedge cfm\ (capteur2)) = 33.33$
$\sigma_T(cfm\ (capteur1) \wedge cfm\ (vanne2)) = (1\ \ 1\ \ 1)$	$\mu_T^c(cfm\ (capteur1) \wedge cfm\ (vanne2)) = 33.33$

TAB. 2 – Mesures de coïncidence des défauts à l'interaction 1

Les mesures de coïncidence montre que le mode de défaut $cfm\ (vanne1)$ est le plus probable.

Probabilité des diagnostics

Le modèle FIS provenant de l'analyse de risques, les probabilités des modes de défaillance sont en général connues. Cette information peut être utilisée pour classer les diagnostics.

Propriété 4. (Probabilité d'un diagnostic) : La probabilité d'un diagnostic est définie à partir des probabilités des modes qui le composent :

$$P(\{mode_1, \ldots, mode_n\}) = P(mode_1) \times \ldots \times P(mode_n)$$

Notons que la propriété (4) est donnée avec l'hypothèse que les modes $\{mode_1, \ldots, mode_n\}$ sont indépendants.

Conclusions

Dans cette partie, nous avons rappelé les principes générales d'un problème de diagnostic logique. Ceci donne une vue d'ensemble sur les outils existants que nous allons utiliser ensuite.

Tout d'abord, l'approche de Reiter (Reiter [1987]) est rappelée. Cette approche permet de calculer tous les diagnostics possibles à partir des symptômes qui sont formulés sous forme des disjonctions de modes. Il permet de calculer des défauts multiples qui peuvent apparaître en même temps dans le système. Aucune hypothèse d'exonération n'est exigées. Ensuite, la mesure de coïncidence et la probabilité des modes sont utilisées comme les outils pour classer les diagnostics. La mesure de coïncidence repose sur la distances Hamming entre la signature effective des tests et la signature théorique des défaut et il n'exige pas l'information sur la fiabilité des éléments du systèmes. A côté de la mesure de coïncidence, la probabilité permet de prendre en compte la fiabilité des diagnostics.

Annexe B

Conception des Sous-Systèmes Testables par la méthode structurelle

Cette annexe résume les résultats présentés dans (Ploix *et al.* [2005b]; Yassine [2008]; Ploix *et al.* [2009, 2010]) sur la méthode structurelle pour la conception des sous-systèmes testables.

La génération de tests de détection comporte généralement deux étapes. La première étape s'appuie sur l'ensemble des contraintes pour générer tout d'abord des sous-systèmes testables (sous entendu, sous-systèmes de contraintes). Dans la deuxième étape, les tests de détection sont générés en partant des sous-systèmes de contraintes. Cette étape consiste à éliminer les variables non-mesurées pour obtenir des relations calculables à partir de mesures. Dans cette annexe, nous allons présenter l'étape de génération de sous-systèmes testables. Dans ce but, nous allons rappeler tout d'abord certaines notions.

Combinaison des contraintes

Soit une relation $K(V)=0$, un ensemble de variables $V^+ \subset V$ qui peuvent être déduites par des fonctions $\forall v \in V^+, \exists K_v : dom(V\backslash\{v\}) \mapsto dom(v)$. On appelera la *structure de contrainte associée* à $K(V)=0$: $\llcorner V^-, V^+ \lrcorner$ avec $V^- = V\backslash V^+$. Les variables de V^+ seront dites déductibles pour la contrainte $K(V)=0$, et V^- non-déductibles pour la contrainte. On notera $var^+(\llcorner V^-, V^+ \lrcorner) = V^+$, $var^-(\llcorner V^-, V^+ \lrcorner) = V^-$ et $var(\llcorner V^-, V^+ \lrcorner) = V^- \cup V^+$.

En algèbre relationnel, si toutes les variables sont déductibles, l'élimination d'une variable v entre deux contraintes $(K_i=0)$ et $(K_j=0)$ peut être représentée par une projection d'une équijointure suivant v entre $(K_i=0)$ et $(K_j=0)$. On note $K_i \bowtie_v K_j = \Pi_{(var(K_i)\cup var(K_j)\backslash\{v\})}(K_i \bowtie_{K_i.v=K_j.v} K_j)$. Comme il peut exister des variables non déductibles pour certaines contraintes dans la représentation structuro-comportementale, toutes les variables ne sont pas nécessairement propageables entre deux contraintes. Les deux définitions suivantes ont été données dans (Yassine [2008]; Ploix *et al.* [2009, 2010]) :

Définition .0.4. *Soient s_1 et s_2 deux structures de contraintes associées respectivement à $(K_1 = 0)$ et à $(K_2 = 0)$. La propagation d'une variable v entre s_1 et s_2 est possible seulement si $v \in var(s_1) \cap var(s_2)$ et si v est déductible dans au moins une structure de contrainte. Si cette condition est satisfaite, v est qualifiée de* propageable *entre s_1 et s_2. Par extension, v est aussi dit* propageable *entre $(K_1 = 0)$ et $(K_2 = 0)$.*

Soient $V^+ = var^+(s)$ et $V^- = var^-(s)$ respectivement l'ensemble des variables déductibles et non-déductibles impliquées dans la structure s d'une contrainte K, une extension de l'opérateur jointure aux contraintes qui peuvent contenir des variables non déductibles peut alors est introduite.

Définition .0.5. *Soient s_1 et s_2 deux structures, avec $V_1^+ = var^+(s_1)$, $V_1^- = var^-(s_1)$, $V_2^+ = var^+(s_2)$ et $V_2^- = var^-(s_2)$. L'opérateur* jointure*, noté $\bowtie_v$, avec v une variable propageable entre s_1 et s_2, est défini dans les deux situations suivantes :*

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

Si une formule $s_1 \bowtie_v s_2$ satisfait un des points précédents, elle est qualifiée d'évaluable.

Le théorème suivant montre comment cet opérateur jointure étendu peut être utilisé pour déduire la structure de contraintes résultant d'éliminations de variables.

Théorème .0.1. *Soient $(K_1 = 0)$ et $(K_2 = 0)$ deux contraintes et s_1, s_2 leurs structures respectivement associées. Une structure englobante de la contrainte résultant de l'élimination d'une variable v propageable entre $(K_1 = 0)$ et $(K_2 = 0)$ est donnée par $s_1 \bowtie_v s_2$.*

Génération des formules de propagation

Les propagations consécutives peuvent aboutir à un sous-système testable qui est modélisé par une formule de propagation comme : $f = (((s_1 \bowtie_{v_1} s_2) \bowtie_{v_2} s_3) \bowtie_{v_3} (s_4 \bowtie_{v_4} s_2))$. Une formule est composée de sous-formules reliées par l'opérateur jointure étendu et où les opérandes sont des structures. Grâce à cet opérateur, une formule de propagation f peut être évaluée sous la forme d'une structure de contrainte : $s(f)$.

Les définitions suivantes présentent des concepts utiles à la génération de tests :

Définition .0.6. *Le support d'une formule $f \in \mathscr{F}$, noté $\sigma(f)$, est l'ensemble de toutes les structures apparaissant dans la formule.*

Définition .0.7. *Le degré d'une formule $f \in \mathbb{F}$, noté $d(f)$, est égal au nombre de fois où l'opérateur jointure étendu apparaît dans la formule : il représente le nombre de propagations élémentaires et donne une idée de la complexité d'une formule.*

Définition .0.8. *Deux formules f_1 et f_2 sont comparables si $\sigma(f_1) = \sigma(f_2)$ et si $var(s(f_1)) = var(s(f_2))$, c'est-à-dire qu'elles comportent les mêmes contraintes et les mêmes variables. On note par $f_1 \sim f_2$.*

Une variable ne peut être instanciée qu'une seule fois durant les propagations. Dans certaines formules, une variable peut apparaître plusieurs fois et donc être instanciée de plusieurs manières différentes. Dans ces situations, il existera une formule plus simple où chaque variable n'aura été instanciée qu'une seule fois.

Définition .0.9. *Une formule f_1 est plus simple qu'une autre formule f_2 si :*
- *$\sigma(f_1) \subset \sigma(f_2)$ et $var(s(f_1)) \subset var(s(f_2))$*
- *ou si $f_1 \sim f_2$ et $d(f_1) < d(f_2)$*
- *ou si $f_1 \sim f_2$ et $d(f_1) = d(f_2)$ et $var^+(s(f_1)) \supset var^+(s(f_2))$*

On note : $f_1 \prec f_2$ et on dira que f_2 surestime f_1.

Une formule de propagation testable est minimale pour un ensemble de formules F si il n'existe pas de formule plus simple que f dans F. Une telle formule est appelée formule de propagation testable minimale (FPTM) sur F.

Combinaisons des formules de propagation

Le nombre de formules d'un ensemble de formules F qui contiennent une variable v est nommé l'ordre de v dans F, noté par $n = O_F(v)$. Le nombre de nouvelles formules qui est généré en éliminant v dans F est le nombre de possibilité de choisir deux éléments dans n, il est calculé par

$$C_n^2 = \frac{n!}{2 * (n-2)!}$$

Par exemple, l'application de l'opérateur $\bowtie_v$ à deux formules f_1 et f_2, conduit à la nouvelle formule : $f_1 \bowtie_v f_2$. Si cette formule est évaluable, f_1 et f_2 sont retirés des formules à combiner. Le nombre de formules est par conséquent réduit de 1. De la même manière, l'application de l'opérateur $\bowtie_v$ aux trois formules f_1, f_2 et f_3 conduit à trois nouvelles formules : $f_1 \bowtie_v f_2$, $f_1 \bowtie_v f_3$ et $f_2 \bowtie_v f_3$. Le nombre de formule reste constant. Si l'on applique l'opérateur $\bowtie_v$ à quatre formules f_1, f_2, f_3 et f_4, on aura au plus six nouvelles formules : $f_1 \bowtie_v f_2$, $f_1 \bowtie_v f_3$, $f_1 \bowtie_v f_4$, $f_2 \bowtie_v f_3$, $f_2 \bowtie_v f_4$ et $f_3 \bowtie_v f_4$.

La figure 2 montre le lien entre le nombre de formules obtenu avant et après l'application des opérateurs jointures. Pour réduire la complexité, il est préférable de commencer par appliquer l'opérateur jointure aux variables ayant l'ordre le plus faible afin d'éviter l'explosion du nombre de formules à combiner. Ce constat motive la méthode de génération de sous-systèmes testables présentée ensuite.

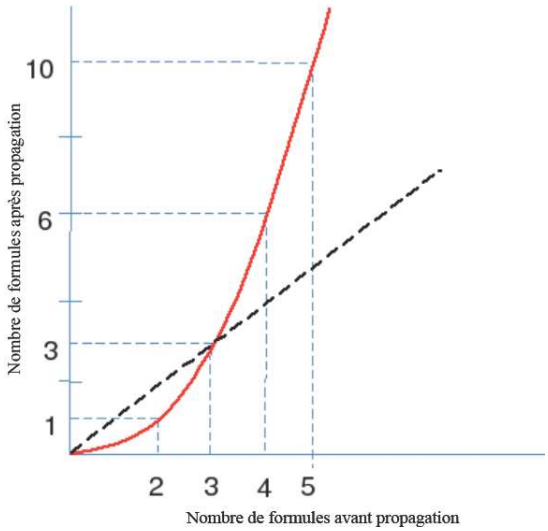


FIG. 2 – Influence de l'opérateur jointure sur le nombre de formules

Génération des sous-systèmes testables

Pour générer les sous-systèmes testables, toutes les formules de propagation testables minimales doivent être trouvées parce que les sous-systèmes testables correspondent aux supports FPTM. Le principe de génération est d'éliminer les variables consécutivement jusqu'à ce que les FPTM soient trouvées.

Les ensembles de formules sont représentés par la lettre F et les structures correspondantes par $s(F)$. Soit F_0 l'ensemble initiale des formules. Avant de commencer les éliminations, certaines formules qui contiennent les variables d'ordre 1 doivent être retirées afin de réduire la complexité. En effet, aucune propagation ne peut être effectuée pour les variables d'ordre 1 dans F_0. Ceci est une première étape de nettoyage.

L'ensemble de formules nettoyé est noté F_1. Comme cela a été évoqué précédemment, les variables avec l'ordre le plus faible sont éliminées d'abord. Soit v, l'une de ces variables. Toutes les formules où v apparaît sont sélectionnées et, en utilisant l'opérateur jointure étendu, de nouvelles formules évaluables sont générées et, si elles ne surestiment pas d'autres formules, elles sont ajoutées à l'ensemble courant des formules à combiner. Les formules contenant v utilisées sont alors retirées de l'ensemble courant des formules à combiner. Cette procédure est répétée jusqu'à ce que toutes les variables aient été éliminées. Les formules restantes correspondent alors à des structures vides (toutes les variables sont éliminées) : les FPTM. La procédure est résumée par l'algorithme suivant :

F_1 : **un ensemble de formules nettoyé** $F \leftarrow F_1$
Tant que $(var(s(F)) \neq \emptyset)$ **faire**
  Sélectionner $v \in var(s(F))$ **tel que** $o_F(v) \leq o_F(v_i), \forall v_i \in var(s(F))$
  $F' \leftarrow \{f / v \in var(s(f))\}$
  $F'' \leftarrow \{f_i \bowtie_v f_j ; (f_i, f_j) \in F'^2, i \neq j, f_i \bowtie_v f_j \, evaluable\}$
  $F \leftarrow (F \backslash F') \cup F''$
  $F \leftarrow F \backslash \{f \in F ; \exists f_i \in F, f_i \neq f, f \succ f_i\}$
Fin Tq
Retourner F

Pour réduire le nombre de propagations, les variables auxquelles sont associées des contraintes terminales peuvent être éliminées exclusivement avec ces contraintes terminales (définition 2.1.3, page 27). L'algorithme précédent peut aussi être utilisé mais quand une variable v est impliquée dans une structure terminale, l'opérateur jointure $\bowtie_v$ est appliqué, d'une part, entre les structures terminales et les autres structures impliquant v, et d'autre part, entre les structures terminales elles-mêmes si plusieurs d'entre elles contiennent v : cela correspond à la notion de redondance matérielle. Les FPTM résultantes sont appelées *FPTM de base*.

Exemple

Nous revenons à l'exemple 14 (page 105) pour illustrer comment les opérateurs jointures sont appliqués pour chercher des sous-systèmes testables à partir de l'ensemble des modèles élémentaires. Les structures de contraintes associées aux modèles élémentaires (représentés sur la figure 4.6) sont données par :

- $(ok(I_1), k_1)$ devient $s(f_1) = \llcorner\{\}, \{L, V, C\}\lrcorner$
- $(ok(I_2), k_2)$ devient $s(f_2) = \llcorner\{\}, \{C\}\lrcorner$
- $(ok(I_3), k_3)$ devient $s(f_3) = \llcorner\{\}, \{L\}\lrcorner$
- $(ok(I_4), k_4)$ devient $s(f_4) = \llcorner\{\}, \{L, V, C\}\lrcorner$
- $(ok(I_5), k_5)$ devient $s(f_5) = \llcorner\{\}, \{V, D\}\lrcorner$
- $(ok(I_8), k_8)$ devient $s(f_8) = \llcorner\{\}, \{D\}\lrcorner$
- $(ok(I_9), k_9)$ devient $s(f_9) = \llcorner\{\}, \{V, P, C, L_{blanc}\}\lrcorner$
- $(ok(I_{10}), k_{10})$ devient $s(f_{10}) = \llcorner\{\}, \{P\}\lrcorner$
- $(ok(I_{14}), k_{14})$ devient $s(f_{14}) = \llcorner\{\}, \{L_{blanc}\}\lrcorner$
- $(ok(I_{15}), k_{15})$ devient $s(f_{15}) = \llcorner\{\}, \{V, P, L_{jaune}, C\}\lrcorner$
- $(ok(I_{17}), k_{17})$ devient $s(f_{17}) = \llcorner\{\}, \{L_{jaune}\}\lrcorner$

En utilisant l'algorithme présenté précédemment pour relier les formules par les opérateurs jointures, les FPTM de base sont données par :

- $SST_1 = ((((f_{15} \bowtie_P f_{10}) \bowtie_{L_{jaune}} f_{17}) \bowtie_C f_2) \bowtie_V (((f_9 \bowtie_P f_{10}) \bowtie_C f_2) \bowtie_{L_{blanc}} f_{14}))$
- $SST_{02} = (((f_1 \bowtie_L f_3) \bowtie_C f_2) \bowtie_V (((f_9 \bowtie_P f_{10}) \bowtie_C f_2) \bowtie_{L_{blanc}} f_{14}))$
- $SST_{03} = (((f_1 \bowtie_L f_3) \bowtie_C f_2) \bowtie_V (((f_{15} \bowtie_P f_{10}) \bowtie_{L_{jaune}} -f_{17}) \bowtie_C f_2))$
- $SST_{04} = (((f_4 \bowtie_L f_3) \bowtie_C f_2) \bowtie_V (((f_9 \bowtie_P f_{10}) \bowtie_C f_2) \bowtie_{L_{blanc}} f_{14}))$
- $SST_{05} = (((f_4 \bowtie_L f_3) \bowtie_C f_2) \bowtie_V (((f_{15} \bowtie_P f_{10}) \bowtie_{L_{jaune}} f_{17}) \bowtie_C f_2))$
- $SST_{06} = ((f_5 \bowtie_D f_8) \bowtie_V (((f_9 \bowtie_P f_{10}) \bowtie_C f_2) \bowtie_{L_{blanc}} f_{14}))$
- $SST_{07} = (((f_4 \bowtie_L f_3) \bowtie_C f_2) \bowtie_V ((f_1 \bowtie_L f_3) \bowtie_C f_2))$
- $SST_{08} = ((f_5 \bowtie_D f_8) \bowtie_V (((f_{15} \bowtie_P f_{10}) \bowtie_{L_{jaune}} f_{17}) \bowtie_C f_2))$
- $SST_{09} = ((f_5 \bowtie_D f_8) \bowtie_V ((f_1 \bowtie_L f_3) \bowtie_C f_2))$
- $SST_{10} = ((f_5 \bowtie_D f_8) \bowtie_V ((f_4 \bowtie_L f_3) \bowtie_C f_2))$

Dans ce travail, nous allons utiliser le logiciel DXLab (développé au laboratoire G-SCOP) pour générer les sous-systèmes testables. Pour décrire structurellement les modèles élémentaires impliqués dans l'exemple précédent, les instructions introduites dans DXLab sont :

$k_1 = |\{-\},\{L,V,C\}|\ models\ I_1$
$k_2 = |\{-\},\{C\}|\ models\ I_2$
$k_3 = |\{-\},\{L\}|\ models\ I_3$
$k_4 = |\{-\},\{L,V,C\}|\ models\ I_4$
$k_5 = |\{-\},\{V,D\}|\ models\ I_5$
$k_8 = |\{-\},\{D\}|\ models\ I_8$
$k_9 = |\{-\},\{V,P,C,L_{blanc}\}|\ models\ I_9$
$k_{10} = |\{-\},\{P\}|\ models\ I_{10}$
$k_{14} = |\{-\},\{L_{blanc}\}|\ models\ I_{14}$
$k_{15} = |\{-\},\{V,P,L_{jaune},C\}|\ models\ I_{15}$
$k_{17} = |\{-\},\{L_{jaune}\}|\ models\ I_{17}$

|{variables non-déductives},{variables déductives}| donne la structure d'un modèle élémentaire correspondant à une contrainte. Les sous-systèmes testables (SST) minimaux (correspondant au FPTM de base) sont trouvés par DXLab :

$SST_{01} : support = k_{15}[I_{15}], k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_2[I_2], k_{17}[I_{17}]$

$Formula = ((((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$

$SST_{02} : support = k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_3[I_3], k_2[I_2], k_1[I_1]$

$Formula = (((k_1 - L - k_3) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$

$SST_{03} : support = k_{15}[I_{15}], k_{10}[I_{10}], k_3[I_3], k_2[I_2], k_1[I_1], k_{17}[I_{17}]$

$Formula = (((k_1 - L - k_3) - C - k_2) - V - (((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2))$

$SST_{04} : support = k_9[I_9], k_{14}[I_{14}], k_{10}[I_{10}], k_4[I_4], k_3[I_3], k_2[I_2]$

$Formula = (((k_4 - L - k_3) - C - k_2) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$

$SST_{05} : support = k_{15}[I_{15}], k_{10}[I_{10}], k_4[I_4], k_3[I_3], k_2[I_2], k_{17}[I_{17}]$

$Formula = (((k_4 - L - k_3) - C - k_2) - V - (((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2))$

$SST_{06} : support = k_9[I_9], k_{14}[I_{14]}, k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2]$

$Formula = ((k_5 - D - k_8) - V - (((k_9 - P - k_{10}) - C - k_2) - L_{blanc} - k_{14}))$

$SST_{07} : support = k_4[I_4], k_3[I_3], k_2[I_2], k_1[I_1]$

$Formula = (((k_4 - L - k_3) - C - k_2) - V - ((k_1 - L - k_3) - C - k_2))$

$SST_{08} : support = k_{15}[I_{15}], k_8[I_8], k_5[I_5], k_{10}[I_{10}], k_2[I_2], k_{17}[I_{17}]$

$Formula = ((k_5 - D - k_8) - V - (((k_{15} - P - k_{10}) - L_{jaune} - k_{17}) - C - k_2))$

$SST_{09} : support = k_8[I_8], k_5[I_5], k_3[I_3], k_2[I_2], k_1[I_1]$

$Formula = ((k_5 - D - k_8) - V - ((k_1 - L - k_3) - C - k_2))$

$SST_{10} : support = k_8[I_8], k_5[I_5], k_4[I_4], k_3[I_3], k_2[I_2]$

$Formula = ((k_5 - D - k_8) - V - ((k_4 - L - k_3) - C - k_2))$

Annexe C

Détection des symptômes par la méthode de propagation de valeurs

Une méthode de propagation de valeur qui permet de resoudre les problèmes de satisfaction de contraintes CSP (Constraint Satisfaction Problem) est utilisée pour détecter des symptômes. Pour expliquer l'intérêt de la méthode de propagation de valeurs en comparaisons avec les méthodes CSP clasique, nous rappelons ici les caractéristiques d'un CSP.

Formellement, un problème de satisfaction de contraintes est défini par un triplet $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ avec $\mathscr{C} = \{k_1, \ldots, k_n\}$ un ensemble fini de contraintes considéré. $\mathscr{X} = var(\mathscr{C}) = \{x_1, \ldots, x_n\}$ est un ensemble fini de variables impliquées dans $\mathscr{C}$. $\mathscr{D} = \{dom(x_1), \ldots, dom(x_n)\}$ sont les domaines qui correspondent aux variables $\{x_1, \ldots, x_n\}$. La résolution d'un CSP consiste à chercher des jeux de valeurs de $\{x_1, \ldots, x_n\}$ tel que toutes les contraintes impliquées dans $\mathscr{C}$ sont satisfaites. Pour résoudre un CSP, certains méthodes proposé dans (Apt [2003]) considère $\{x_1, \ldots, x_n\}$ comme une liste ordonnée des variables. Les valeurs sont affectées successivement à chaque que variable de $\{x_1, \ldots, x_n\}$ en vérifiant les contraintes. La vérification est réalisée soit par la vérification en avance (forward checking) soit par la vérification en arrière (backward checking).

Adaptons ces principes à notre problème de génération automatique de symptômes par propagation de valeurs au sein de SSTs. L'ordre des variables à affecter est choisi automatiquement dans la procédure de propagation quand nous prenons en compte la notion de déductibilité et non-déductibilité d'un variable (Yassine [2008]). Il est possible de déduire les valeurs d'une variable déductible à partir d'une variable non-déductible mais non dans ce sens inverse. La notion de variable déductible / non-déductible peut exister dans les modèles de comportement (souvent des modèle continues) ou dans la fonction de connexion partielle entre deux variables impliquées dans deux modèles élémentaires (2.1.4). Rappelons qu'un SST compte à la fois des contraintes de comportement décrivant des modes des items et aussi des contraintes décrivant les connexions entre les variables des modèles élémentaires. Les contraintes de connexions ne s'associent à aucun mode.

Selon la définition de SST (définition 4.3.2), pour un ensemble de contraintes K impliqué dans un SST_i, il existe toujours une variable v_j tel que nous pouvons atteindre v_j en propageant à partir des valeurs observées (valeurs liées aux variables des contraintes terminales). De cette intuition, nous proposons une méthode pour détecter des symptômes par la propagation de valeurs à partir des contraintes terminales. Cette méthode est basé sur le problème de satisfaction de contrainte $\langle \mathscr{X}', \mathscr{D}', \mathscr{C}' \rangle$, où $\mathscr{C}'$ est modifiable. Dans la procédure de propagation, les contraintes peuvent être ajoutées dans $\mathscr{C}'$, $\mathscr{X}' = var(\mathscr{C}')$. L'ordre d'affectation des variables $x_i \in \mathscr{X}'$ est déterminé automatiquement dans la procédure de propagation. Les différences entre le CSP original $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ et l'approche de propagation de valeurs utilisée sont résumées sur le tableau 3.

CSP original $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$	Approche de propagation de valeur $\langle \mathscr{X}', \mathscr{D}', \mathscr{C}' \rangle$
$\mathscr{C}$ est un ensemble fini $\mathscr{X}$ est une liste ordonnée	$\mathscr{C}'$ est modifiables $\mathscr{X}'$ est choisi automatiquement dans la propagation

TAB. 3 – Distinction entre le CSP original et l'approche de propagation de valeur

La méthode de propagation de valeurs consiste à restreindre les domaines de valeurs $\{dom(x_1), \ldots, dom(x_n)\}$ aux valeurs qui satisfont l'ensemble des contraintes $\mathscr{C}'$. Si des domaines de valeurs vides sont trouvés, cela veut dire qu'il y a un conflit entre le modèle de référence et les données collectées par l'expert et donc que le *SST* correspondant doit être étiqueté N sur le graphe de priorité des test. On dit qu'un jeu de valeurs $\{v_1 \in dom(x_1), \ldots, v_n \in dom(x_n)\}$ est une solution de $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ s'il satisfait chaque contrainte $(k) \in \mathscr{C}$.

La méthode de propagation de valeurs peut être appliquée pour des modèles discrets, continus, ou des modèles où le domaine des variables sont des intervalles, etc. Ceci dépend de la capacité du solveur. Pour illustrer le problème de détection automatique des symptômes à partir des SSTs, nous présentons *une méthode de propagation de valeur* pour le modèle discret où les variables sont binaires ou entière ou symboliques. Supposons que certaines variables impliquées dans les contraintes soient non-déductibles. Cette approche est basée sur *la méthode de vérification en avance (Forward checking)* présentée dans (Apt [2003]) pour résoudre un CSP.

Propagation de valeurs par la vérification en avant (Forward checking)

Dans cette partie, nous allons nous concentrer sur chaque SST_i donné par la méthode structurelle de génération de test. Nous allons propager des valeurs dans chaque SST_i en partant des contraintes terminales (chaque contrainte terminal est une valeur s'associant à une variable)

pour vérifier la consistance du test. Pour reformuler une méthode de propagation de valeur se basant sur les techniques de résolution de CSP, rappelons en les principales notions.

Définition .0.10. *(affectation) Soit un CSP* $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ *avec* $\mathscr{C} = \{k_1, \ldots, k_n\}$ *et* $\mathscr{X} = var(\mathscr{C}) = \{x_1, \ldots, x_n\}$. *Une affectation est un ensemble de couples variables/valeurs* $A = \{x_1 \leftarrow v_1, \ldots, x_r \leftarrow v_r\}$ *tel que* $0 \leq r \leq n$ *et* $\forall i \in [1...r] v_i \in dom(x_i)$.

Chaque affectation $A = \{x_1 \leftarrow v_1, \ldots, x_r \leftarrow v_r\}$ *correspond à un ensemble de singletons* $\{E_1 = \{v_1\}, \ldots, E_r = \{v_r\}\}$ *avec* $E_i \subseteq dom(x_i)$.

Une affectation est partielle si elle ne concerne qu'une partie des variables de $\mathscr{X}$, et totale si elle concerne toutes les variables de $\mathscr{X}$.

Une affectation A est valide par rapport à $\mathscr{C}$ si la relation définie dans chaque contrainte k_i est vérifiée pour les valeurs des variables affectées dans A.

Définition .0.11. *Une solution d'un CSP* $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ *est une affectation totale valide de ce CSP.*

Principe de propagation de valeurs

Rappelons que la propagation de valeurs au sein de chaque SST_i est considéré comme un CSP $\langle \mathscr{X}', \mathscr{D}', \mathscr{C}' \rangle$ où $\mathscr{C}'$ est modifiable. Soit $\mathbb{K}$ un ensemble des contraintes impliquées dans SST_i, nous avons $\mathscr{C}' \subseteq \mathbb{K}$. Initialement, $\mathscr{C}'$ est l'ensemble des contraintes qui sont liées aux variables mesurées impliquées dans les contraintes terminales. La procédure de propagation de valeurs au sein de SST_i commence. La procédure de propagation de valeurs est une procédure itératif en deux phases : fragmentation (spliting) et propagation. Cette procédure est présentée par la figure (3). Décrivons tout d'abord le rôle de chaque tâche.

- **La fragmentation** (représentée par un arc normal dans la figure 3) est suivi par la propagation. Cette tâche consiste à choisir une variable non-affectée dont le domaine n'est pas un singleton. Le domaine de la variable choisie sera fragmenté ici en singletons. Puis chaque valeur sera affectée successivement à cette variable. Nous allons distinguer ici trois types de variable :
  - **variable passée :** Les variables qui sont déjà affectées sont appelées des *variables passée*. Le domaine de chaque variable passée a la forme d'un singleton.
  - **Variable courante :** La variable qui est choisie pour cette fragmentation est appelé *variable courante*.
    Une variable x_i est choisie si $\exists x_j$ tel que $\{x_i, x_j\} \in var(k)$ x_j est non-affecté et x_j est déductible de x_i via la contrainte k.
    Si plusieurs variables x_i sont disponibles, nous choisissons de fragmenter et d'affecter la variable dont le domaine a le cardinal le plus petit.

- **variable future :** Soit $\mathscr{C}' \subseteq \mathbb{K}$ un ensemble de contraintes qui contient les variables courants et les variables passées. Les variables impliquées dans $\mathscr{C}'$ à l'exception des *variables passées* et de la *variable courante* sont appelées *variables futures*. Durant la propagation, les domaines des *variables futures* seront restreints par l'affectation formée des *variables passées* et de la *variable courante*. Remarquons qu'à cette étape l'ensemble $\mathscr{C}'$ est mis à jours. Les nouvelles contraintes de $\mathbb{K}$ qui sont liées avec la variable courante sont ajoutées dans $\mathscr{C}'$.

– **La propagation** (représentée par un arc pointillé dans la figure 3) suit la fragmentation. Supposons que la fragmentation précédente donne une affectation partielle $A = \{x_1 \leftarrow v_1, \ldots, x_r \leftarrow v_r\}$ qui correspond à un ensemble de singleton $(E_1 = \{v_1\}, \ldots, E_r = \{v_r\})$. La propagation via des contraintes consiste à vérifier les singletons (les variables passés et la variable courante) avec des contraintes associées. Cette propagation permet aussi de supprimer du domaine des variables futures, les valeurs qui ne sont pas compatibles avec l'affectation partielle considérée. Pour le distinguer du domaine initial $D_i = dom(x_i)$ de la variable x_i, le domaine affiné d'une variable non affectée obtenu après une propagation est noté par E_i avec $E_i \subseteq D_i$.

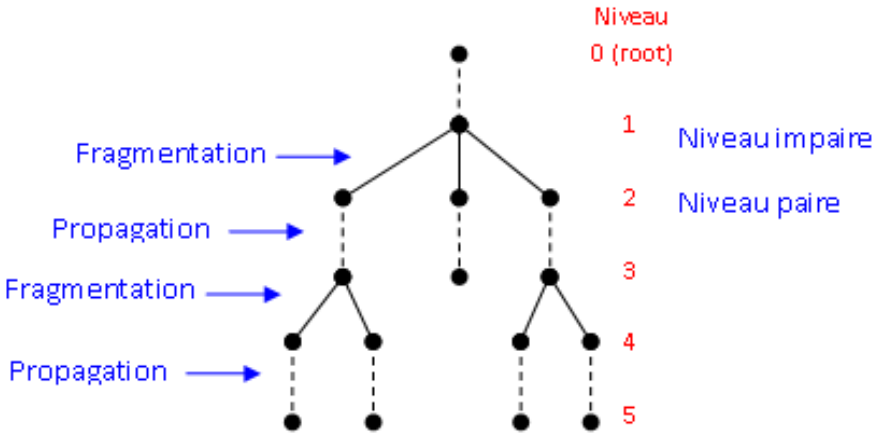


FIG. 3 – Arbre de propagation pour un CSP

Arbre de propagation

La propagation de valeurs peut être réalisée au moyen d'un arbre de propagation (figure 3). Soit un problème de satisfaction de contrainte initial $\mathscr{P}\ \langle \mathscr{X}', \mathscr{D}', \mathscr{C}' \rangle$ avec $\mathscr{X}' = var(\mathscr{C}') = \{x_1, \ldots, x_n\}$, un ensemble de variables impliquées dans $\mathscr{C}'$. Les nouvelles contraintes peuvent être ajoutées dans $\mathscr{C}'$ dans la procédure de propagation. L'arbre de propagation pour la méthode de propagation de valeurs est défini par :

– Chaque noeud est étiqueté par un ensemble $(x_1 \in \{v_1\}, \ldots, x_i \in \{v_i\}, x_{i+1} \in E_{i+1}, \ldots, x_n \in E_n)$ où $\{x_1, \ldots, x_{i-1}\}$ sont des variables passées, x_i est la variable cou-

rante et $\{x_{i+1},\ldots,x_n\}$ sont les variables futures. Chaque élément $x_i \in E_i, i \in [1,\ldots,n]$ est appelé *l'expression du domaine* avec $E_i \subseteq D_i$ et $D_i = dom(x_i)$. E_i est un domaine restreint de D_i après une propagation. La racine est étiquetée par l'ensemble $\{x_1 \in D_1,\ldots,x_n \in D_n\}$. Sur cette arbre (figure 3), chaque noeud obtenu lors d'une itération impaire est un noeud obtenu après une propagation. Chaque noeud obtenu après une itération paire est un noeud obtenu après une fragmentation.
Les noeuds au niveau impaire (après une propagation) qui n'ont pas de branches descendantes sont appelés *feuilles*.

Définition .0.12. *(feuille d'un arbre de propagation) Un noeud* $\{x_1 \in E_1,\ldots,x_n \in E_n\}$ *est une feuille de l'arbre de propagation si :*
$\exists E_k = \emptyset$ *avec* $k \in [i,\ldots,n]$
ou si :
l'ensemble des variables futures est vide.

– A chaque itération paire, chaque noeud prend la forme

$$(x_1 \in \{v_1\},\ldots,x_i \in \{v_i\},x_{i+1} \in E_{i+1},\ldots,x_n \in E_n)$$

Ce noeud a une descendance

$$(x_1 \in E'_1,\ldots,x_i \in E'_i,x_{i+1} \in E'_{i+1},\ldots,x_n \in E'_n)$$

où $E'_k \subseteq E_k$ pour $k \in [i,\ldots,n]$ obtenu par la vérification des contraintes contenant $\{x_1,\ldots,x_i\}$ ou par la propagation de valeurs à partir des singletons $(x_1 \in \{v_1\},\ldots,\{x_i \in \{v_i\})$ via les contraintes $\mathscr{C}$. E'_k peut être un ensemble vide.
– A chaque itération impaire, chaque noeud prend la forme

$$(x_1 \in \{v_1\},\ldots,x_i \in \{v_i\},x_{i+1} \in E_{i+1},\ldots,x_n \in E_n)$$

Si ce noeud n'est pas une feuille, il y a m descendantes de forme :

$$(x_1 \in \{v_1\},\ldots,x_i \in \{v_i\},x_{i+1} \in \{v_{i+1}\},\ldots,x_n \in E_n)$$

où m est le nombre d'éléments impliqués dans E_{i+1}. Ce noeud est obtenu par l'affectation $x_{i+1} \leftarrow v_{i+1}$ avec $v_{i+1} \in dom(x_{i+1})$
Si un noeud issu d'une itération impaire n'est pas une feuille, la tâche de fragmentation sera appliquée au noeud courant.

Une feuille $\{x_1 \in E_1,\ldots,x_n \in E_n\}$ est appelée un noeud réussite si chaque ensembles E_i est un singleton avec $i \in [1,\ldots,n]$. Dans le cas contraire, on parle de noeud échec. Chaque noeud réussi qui correspond à une affectation totale et valide est une solution du CSP.

Exemple

Pour illustrer l'approche de génération automatitique de symptômes, nous prenons ici l'exemple simple d'un circuit électrique (figure 4). Dans cet exemple, nous n'allons pas nous concentrer sur le processus de diagnostic interactif mais sur la partie de génération de symptômes. Pour cette raison, nous enrichissons le modèle du système pour qu'il peut apparaisse plusieurs tests et regardons directement le système au niveau le moins abstrait.

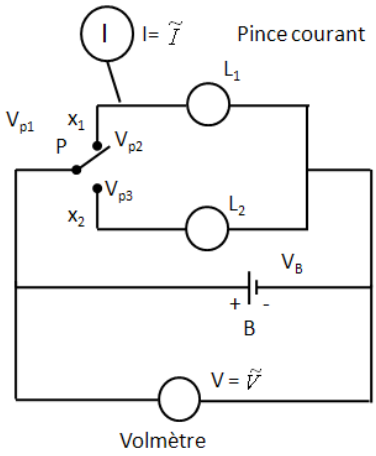


FIG. 4 – Exemple d'un circuit électrique

Ce circuit se compose de :
- un commutateur modélisé par le tableau de comportement 5
- une lampe 1 modélisé par le tableau de comportement 6
- une lampe 2 modélisé par le tableau de comportement 7
- une batterie modélisé par le tableau de comportement 4
- un voltmètre modélisé par le tableau de comportement 8
- une pince de courant modélisé par le tableau de comportement 9

Modes	V_B (tension)
ok	forte
ok	faible

TAB. 4 – Comportement de la Batterie (k_1)

On utilise tout d'abord la méthode structurelle pour chercher des sous système testable. La structure du système peut être écrite par DXLab (voir annexe B) :

Modes	P (position du commutateur)	V_B (tension)	V_{p2} (tension)	V_{p3} (tension)
ok/cfm	x_1	faible	faible	faible
ok	x_2	faible	faible	faible
ok	x_1	forte	forte	faible
ok	x_2	forte	faible	forte

TAB. 5 – Comportements du Commutateur (k_2)

Modes	V_{p2} (tension)	I (courant électrique)	L_1 (lumière)
ok	1	1	allumé
ok	0	0	éteigne

TAB. 6 – Comportement de la Lampe 1 (k_3)

$k_1 = |\{-\},\{V_B\}|\ models\ Batterie$
$k_2 = |\{-\},\{V_B, V_{p2}, V_{p3}, P\}|\ models\ Commutateur$
$k_3 = |\{-\},\{V_{p2}, L_1, I\}|\ models\ Lampe1$
$k_4 = |\{-\},\{V_{p3}, L_2\}|\ models\ Lampe2$
$k_5 = |\{-\},\{V_B\}|\ models\ Volmetre$
$k_6 = |\{-\},\{I\}|\ models\ Pince$
$k_7 = |\{-\},\{P\}|\ models\ PositionCom$
$k_8 = |\{-\},\{L_1\}|\ models\ Lumiere1$
$k_9 = |\{-\},\{L_2\}|\ models\ Lumiere2$

Les SSTs trouvés par la méthode structurelle sont :

$SST_{01} : support = Commutateur[k_2], Lampe2[k_4], PositionCom[k_7], Batterie[k_1], Lampe1[k_3], Lumiere2[k_9], Pince[k_6], Lumiere1[k_8]$
$Formula = (((((((Commutateur - V_{p3} - Lampe2) - V_{p2} - Lampe1) - L_2 - Lumiere2) - I - Pince) - L_1 - Lumiere1) - P - PositionCom) - V_B - Batterie)$
$SST_{02} : support = Volmetre[k_5], Batterie[k_1]$
$Formula = (Volmetre - V_B - Batterie)$
$SST_{03} : support = Commutateur[k_2], Volmetre[k_5], Lampe2[k_4], PositionCom[k_7], Lampe1[k_3], Lumiere2[k_9], Pince[k_6], Lumiere1[k_8]$
$Formula = (Volmetre - V_B - ((((((Commutateur - V_{p3} - Lampe2) - V_{p2} - Lampe1) - L_2 - Lumiere2) - I - Pince) - L1 - Lumiere1) - P - PositionCom))$

La méthode de propagation de valeurs peut alors être appliquée à chaque SST pour

Modes	V_{p3} (tension)	L_2 (lumière)
ok/cfm	0	absent
ok/cfm	1	présent

TAB. 7 – Comportements de la Lampe 2 (k_4)

Modes	V_B (tension)	$\tilde{V_B}$ (tension)
ok	forte	forte
ok	faible	faible

TAB. 8 – Comportement du Voltmètre (k_5)

chercher des jeux de valeurs candidats. Nous allons prendre ici l'exemple de SST_{03}. La recherche des jeux de valeurs pour SST_{03} correspond à un CSP $\langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ avec $\mathscr{C} = \{k_2, k_5, k_4, k_7, k_3, k_9, k_6, k_8\}$. k_7, k_8, k_9 sont des observateurs humains qui sont représentés par :

- $k_7 : P = \tilde{P}$
- $k_8 : L_1 = \tilde{L_1}$
- $k_9 : L_2 = \tilde{L_2}$

$\mathscr{X} = VAR_{\angle}(\mathscr{C}) = \{V_B, P, I, V_{p2}, V_{p3}, L_1, L_2\}\}$. Notons que $\tilde{V_B}$, $\tilde{P}$, $\tilde{L_1}$, $\tilde{L_2}$ sont les valeurs associés aux variables et n'apparaissent pas dans $\mathscr{X}$.

La recherche des jeux de valeurs par arbre de propagation est donnée par la figure 5. Chaque noeud est étiqueté par un ensemble $(x_1 \in E_1, \ldots, x_n \in E_n)$. Pour simplifier, si $E_i \equiv D_i$ alors $x_i \in E_i$ est noté simplement par x_i sur le schéma.

Les jeux de valeurs trouvés sont :

0. (V_B={faible}, V_{p2}={0}, V_{p3}={0}, P={x1}, I={0}, L_1={éteigne}, L_2={éteigne})

1. (V_B={faible}, V_{p2}={0}, V_{p3}={0}, P={x2}, I={0}, L_1={éteigne}, L_2={éteigne})

2. (V_B={forte}, V_{p2}={0}, P={x2}, V_{p3}={1}, I={0}, L_1={éteigne}, L_2={allumé})

3. (V_B={forte}, V_{p2}={1}, P={x1}, V_{p3}={0}, I={1}, L_1={allumé}, L_2={éteigne})

Dans cette partie, nous avons reformulé *la méthode de vérification en avant* pour un problème de CSP sous forme d'*un méthode de propagation de valeur*. Ceci permet d'améliorer la méthode de *vérification en avance* originale en tenant compte de la notion de déductibilité / non-déductibilité des variables. Cette méthode a été formulée pour un modèle aux variables discrètes avec des variables déductibles et non-déductible peuvent exister. L'extension aux domaines de valeurs continus suit le même principe.

Modes Modes	I (courant électrique)	$\tilde{I}$ (courant électrique
ok ok	1 0	1 0

TAB. 9 – Comportement de la Pince de courant (k_6)

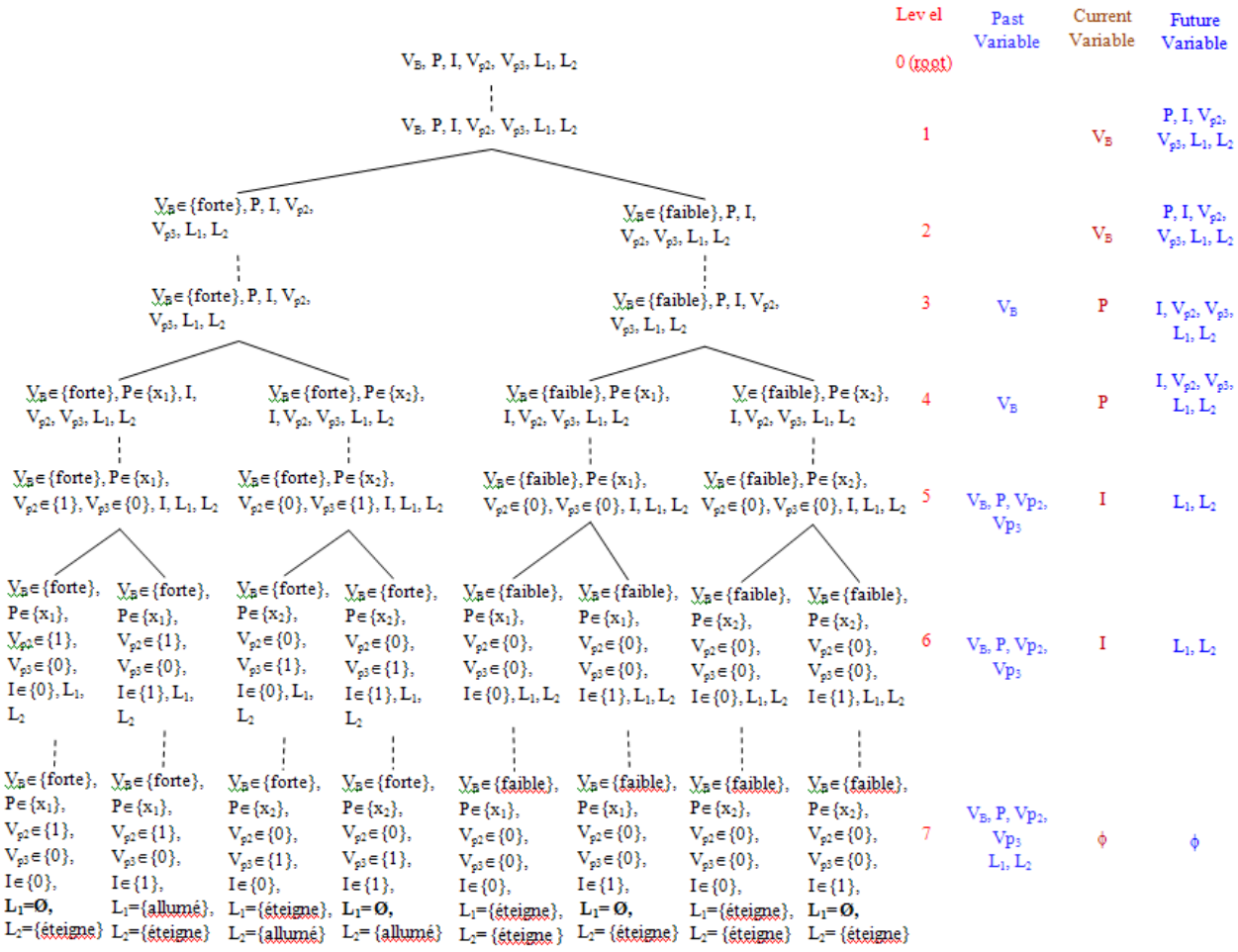


FIG. 5 – Arbre de propagation

Annexe D

Cette annexe présente les résultats du raisonnement floue dans l'analyse diagnostique donnée dans Touaf et Ploix [2004b].

Fuzzification des symptômes

L'idée est de dire que si les contraintes de validité ne sont pas satisfaites, au lieu de considérer que le système se comporte a priori normalement, il est dans un état indéterminé. Cela se traduit par une relation valide uniquement sous hypothèse d'exonération (tous les observables ont été parcourus) :

$$\bigwedge_i ok(item_i) \leftarrow (\mathscr{K}(V)=0) \wedge \left(\mathscr{K}'(V)=0\right) \tag{.0.3}$$

par une contrainte modélisant le doute en cas de non-validité :

$$\neg\left(\mathscr{K}'(V)=0\right) \equiv \text{ doute sur } \bigwedge_i ok(item_i) \tag{.0.4}$$

et par une contrainte correspondant à une observation donnée :

$$\bigwedge_i ok(item_i) \rightarrow (\mathscr{K}(V)=0) \vee \neg\left(\mathscr{K}'(V)=0\right) \tag{.0.5}$$

Il faut donc tester deux ensembles de contraintes. Dans les deux cas, avec l'approche floue, nous considérerons qu'il existe plus de deux valeurs possibles pour la satisfaction (*satisfait* ou *non-satisfait*), mais une infinité qui va de 1 (satisfait) à 0 (non satisfait). Cela correspond au degré d'appartenance flou à *satisfait*. La figure 6 illustre ce point de vue. A l'issu de la phase de test, il existe deux degrés d'appartenance : $\mu_{\mathscr{K}} = \mu(\mathscr{K}(V)=0)$ et $\mu_{\mathscr{K}'} = \mu(\mathscr{K}'(V)=0)$. Nous avons le degré d'appartenance $\mu_{\mathscr{K}} = \mu(\mathscr{K}(V)=0) = 1$ si $\mathscr{K}(V)=0$ est totalement satisfait, et $\mu_{\mathscr{K}} = \mu(\mathscr{K}(V)=0) = 0$ si $\mathscr{K}(V)=0$ est totalement insatisfait. Il est pareil pour $\mu_{\mathscr{K}'} = \mu(\mathscr{K}'(V)=0)$. En utilisant l'interprétation floue $\mu(A \vee B) = min(1, \mu(A)+\mu(B))$, où A et B sont des propositions, nous transposons[1] (.0.5) en :

[1] On rappelle que $A \rightarrow B$ équivaut à $\neg A \vee B$.

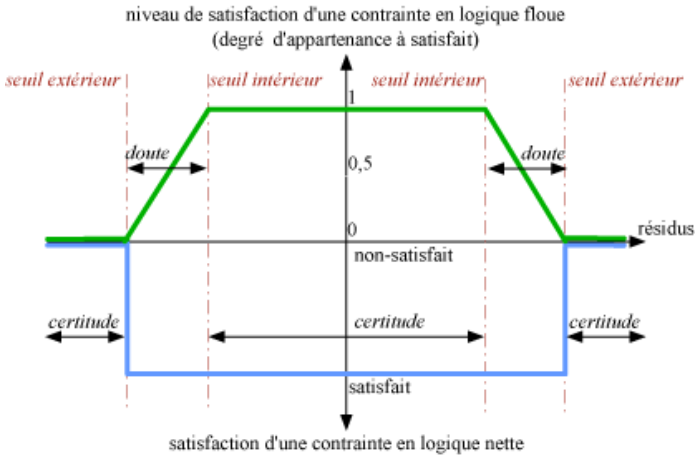


FIG. 6 – Interprétation des résidus dans un contexte flou

$$min\left(1,1-\mu\left(\bigwedge_i ok(item_i)\right)+\mu\left(\mathscr{K}(V)=0\right)+\left(1-\mu\left(\mathscr{K}'(V)=0\right)\right)\right)=1$$

ou encore, en adoptant les notations précédentes :

$$min\left(1,2-\mu\left(\bigwedge_i ok(item_i)\right)+\mu_{\mathscr{K}}-\mu_{\mathscr{K}'}\right)=1$$

Pour satisfaire cette relation, il faut vérifier :

$$\mu\left(\bigwedge_i ok(item_i)\right)\leq 1+\mu_{\mathscr{K}}-\mu_{\mathscr{K}'} \tag{.0.6}$$

Dans le cas où la contrainte de validité est totalement insatisfaite, ou on a $\mu_{\mathscr{K}'}=\mu(\mathscr{K}'(V)=0)=0$. Ceci conduit à un cas de doute où le système est dans un état indéterminé. Nous ne pouvons donc pas décider ni vrai (=1) ni faux (=0). La relation (.0.4) conduit à :

$$(\mu_{\mathscr{K}'}=0)\equiv\left(\mu\left(\bigwedge_i ok(item_i)\right)=0.5\right) \tag{.0.7}$$

Enfin, la relation (.0.3), qui n'est valide que sous hypothèse d'exonération, se transpose en :

$$min\left(1,2-\mu_{\mathscr{K}}-\mu_{\mathscr{K}'}+\mu\left(\bigwedge_i ok(item_i)\right)\right)=1$$

Pour satisfaire cette relation, il faut vérifier :

$$\mu\left(\bigwedge_i ok(item_i)\right) \geq \mu_{\mathscr{K}} + \mu_{\mathscr{K}'} - 1 \tag{.0.8}$$

Le problème est alors de trouver une fonction de fuzzification qui permette de déduire le degré d'appartenance au symptôme $\bigwedge_i ok(item_i)$ et partant des degrés de satisfaction des contraintes $(\mathscr{K}(V) = 0)$ et $(\mathscr{K}'(V) = 0)$. Cette fonction doit vérifier les contraintes (.0.6) et (.0.7) qui sont représentées sur la figure 7. Les parties non-grisées satisfont les contraintes. Les cercles rouges correspondent à des points de passage obligés du fait de la contrainte (.0.4). De plus, si la fonction de fuzzification permet de vérifier (.0.8), elle pourra aussi être utilisée sous hypothèse d'exonération. Nous avons montré que la fonction suivante convenait :

$$\mu(test) = \Gamma(\mu_{\mathscr{K}}, \mu_{\mathscr{K}'}) = \frac{1 + (2\mu_{\mathscr{K}} - 1)\mu_{\mathscr{K}'}}{2} \tag{.0.9}$$

Pour chaque test, cette fonction de fuzzification permet d'évaluer le degré de vérité des symptômes en fonction des degrés de satisfaction des contraintes de comportement et de validité : $\mu\left(\bigwedge_i ok(item_i)\right) = \Gamma(\mu_{\mathscr{K}}, \mu_{\mathscr{K}'})$.

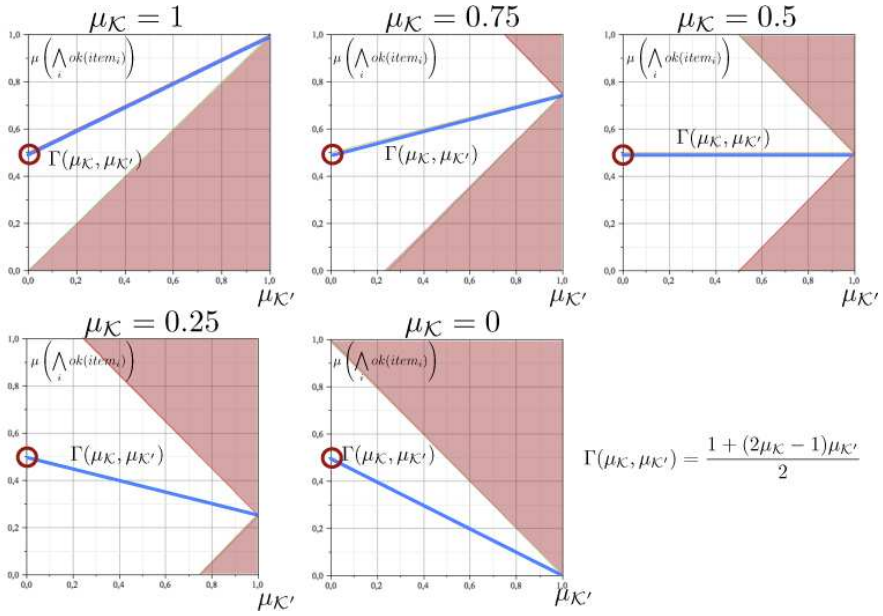


FIG. 7 – Fonction de fuzzification pour les tests flous

En cas d'anomalie, au lieu d'évaluer $\mu\left(\bigwedge_i ok(item_i)\right)$, on va plutôt chercher à évaluer $\mu\left(\bigvee_i cfm\ (item_i)\right) = 1 - \mu\left(\bigwedge_i ok(item_i)\right)$ car dans un cas non multi-modal, on a : $\neg ok(item) = cfm\ (item)$. Il s'ensuit :

$$\mu\left(\bigvee_i cfm\ (item_i)\right) = \mu\left(\neg(\mathscr{K}(V) = 0) \wedge \left(\mathscr{K}'(V) = 0\right)\right) = 1 - \Gamma(\mu_{\mathscr{K}}, \mu_{\mathscr{K}'}) \tag{.0.10}$$

Imaginons maintenant que deux tests de détection aient été effectués. Dans ce cas, deux systèmes de contraintes seront disponibles : $(\mathscr{K}_1(V)=0, \mathscr{K}'_1(V)=0)$ et $(\mathscr{K}_2(V)=0, \mathscr{K}'_2(V)=0)$. La relation (.0.5) devient :

$$\bigwedge_i ok(item_i) \rightarrow \big((\mathscr{K}_1(V)=0) \vee \neg\big(\mathscr{K}'_1(V)=0\big)\big) \wedge \big((\mathscr{K}_2(V)=0) \vee \neg\big(\mathscr{K}'_2(V)=0\big)\big)$$

En cas d'anomalie, on obtient :

$$\neg\big((\mathscr{K}_1(V)=0) \vee \neg\big(\mathscr{K}'_1(V)=0\big)\big) \vee \neg\big((\mathscr{K}_2(V)=0) \vee \neg\big(\mathscr{K}'_2(V)=0\big)\big) \rightarrow \bigvee_i cfm\ (item_i)$$

Ainsi, en utilisant l'opérateur de fuzzification *max* pour $\vee$ afin d'éviter la saturation, on déduit :

$$\mu\left(\bigvee_i cfm\ (item_i)\right) = max(1-\Gamma(\mu_{\mathscr{K}_1}, \mu_{\mathscr{K}'_1}), 1-\Gamma(\mu_{\mathscr{K}_2}, \mu_{\mathscr{K}'_2}))$$

Ce résultat se généralise à l'ordre n et montre que, sans hypothèse d'exonération, le degré d'alarme $\mu\left(\bigvee_i cfm\ (item_i)\right)$ est soit constant soit croissant. La figure 8 illustre ce résultat sur un exemple où les nouvelles observations sont inscrites dans une série temporelle.

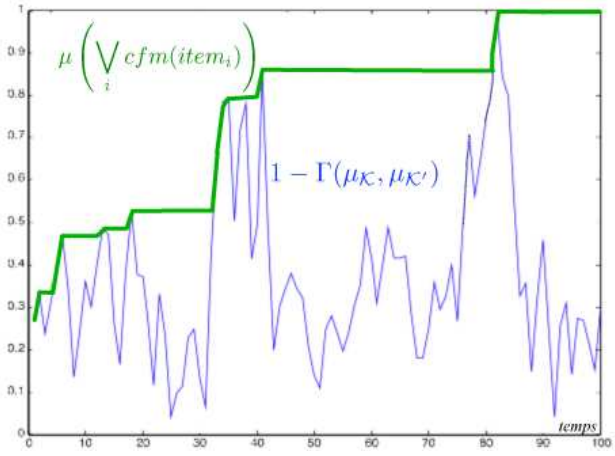


FIG. 8 – Le degré d'alarme dans un contexte flou formel

Raisonnement diagnostique flou

En phase de détection, un degré de vérité $\mu_{test} = \mu\left(\bigwedge_i ok(item_i)\right)$ est évalué pour chaque test. La phase d'analyse diagnostique peut donc démarrer. Mais l'analyse diagnostique formelle peut-elle être transposée de la logique nette à la logique floue ? Plusieurs situations

peuvent se produire. Soit $\mathbb{T}$, l'ensemble des tests que l'on partitionne en $\mathbb{T} = \mathbb{T}^{satisfait} \cup \mathbb{T}^{insatisfait} \cup \mathbb{T}^{douteux}$ avec $\mathbb{T}^{satisfait} = \{test; \mu(test) = 1\}$, $\mathbb{T}^{insatisfait} = \{test; \mu(test) = 0\}$ et $\mathbb{T}^{douteux} = \{test; \mu(test) \in]0,1[\}$. On considère qu'un test douteux peut être soit satisfait soit insatisfait. Le principe de résolution est d'examiner toute la combinatoire possible sur les tests douteux est, pour chaque cas, de collecter les diagnostics. Parmi cet ensemble de diagnostics, seuls les diagnostics minimaux sont conservés car ils sont générateurs des autres diagnostics. Examinons les 3 situations différentes.

La première correspond au cas où $\mathbb{T} \equiv \mathbb{T}_{satisfait}$. Cette situation est similaire à celle de la logique nette : il n'y a aucune raison de calculer des diagnostics car aucune anomalie n'a été révélée.

La seconde situation est celle où $\mathbb{T}^{insatisfait} \neq \emptyset$. Les diagnostics doivent alors pouvoir expliquer non seulement ces tests mais aussi les tests douteux $\mathbb{T}^{douteux}$ qui révèlent une anomalie possible. Appelons $\mathbb{D}^{insatisfait} = \{D_i^{insatisfait}\}$, les diagnostics expliquant les tests $\mathbb{T}^{insatisfait} \neq \emptyset$. Pour expliquer la combinatoire des tests $\mathbb{T}^{douteux}$, chaque diagnostic $D_i^{insatisfait}$ doit être complété par des modes *cfm* supplémentaires. Or, les diagnostics résultant de ce complément sont nécessairement non minimaux puisqu'ils contiennent $D_i^{insatisfait}$. Par conséquent, dans la situation où $\mathbb{T}^{insatisfait} \neq \emptyset$, les diagnostics minimaux sont déduits uniquement des tests $\mathbb{T}^{insatisfait}$: $\{D_i\} = \{D_i^{insatisfait}\}$ ou encore, écrit différemment, $\mathbb{D} = \mathbb{D}^{insatisfait}$. Les diagnostics peuvent être calculés mais il reste à évaluer leur degré de vérité. Le principe est d'évaluer le degré de vérité de ce qui a induit le diagnostic. Dans cette situation, les diagnostics sont induits par les tests $\mathbb{T}^{insatisfait}$ dont le degré de non satisfaction est donné par : $1 - \mu(test) = 1$. Par conséquent, puisque tous les niveaux de non-satisfaction des tests ayant générés les diagnostics $\mathbb{D}^{insatisfait}$ valent 1, on déduit que :

$$\forall D_i \in \mathbb{D}^{insatisfait}, \mu(D_i) = 1 \tag{.0.11}$$

Ainsi, s'il existe des tests avec un $\mu(test) = 0$, les diagnostics doivent être calculés uniquement à partir des tests vérifiant $\mu(test) = 0$ et leur degré de vérité est de 1.

La dernière situation est celle où $\mathbb{T}^{insatisfait} = \emptyset$ et $\mathbb{T}^{douteux} \neq \emptyset$. En considérant toute la combinatoire possible pour les tests de $\mathbb{T}^{douteux}$, il apparaît que les diagnostics minimaux correspondent aux cas où un seul test douteux est insatisfait[2]. Les diagnostics correspondent à des modes *cfm* simples : $\mathbb{D}^{douteux} = \bigcup_{test \in \mathbb{T}^{douteux}} \bigcup_{mode_i \in Expl(test)} mode_i$. Pour évaluer le degré de vérité d'un diagnostic de ce type, il faut là encore évaluer le degré de vérité de ce qui a conduit à ce diagnostic. Si l'on considère un diagnostic *cfm* $(item_i)$, il appartient à l'ensemble des diagnostics $\mathbb{D}^{douteux}$ parce que certains tests le contiennent. En utilisant *max* comme opérateur de fuzzification pour $\vee$, son degré de vérité sera donc :

$$\mu(cfm\ (item_i)) = max_{test \in \mathbb{T}^{douteux}} \left(1 - \Gamma(\mu_{\mathscr{K}_{test}}, \mu_{\mathscr{K}'_{test}})\right) \tag{.0.12}$$

[2]Il n'y a aucun diagnostic si tous les tests sont satisfaits.

La fonction Γ est utilisable lorsque l'hypothèse d'exonération est faite. Il est donc possible de s'en servir pour évaluer une distance entre la signature effective définie par $\forall test_i \in \mathbb{T}, (\sigma^*_{\mathbb{T}})_i = 1 - \Gamma(\mu_{\mathscr{K}_{test_i}}, \mu_{\mathscr{K}'_{test_i}})$ [3], et la signature théorique d'un diagnostic. Soit un diagnostic D_j. La signature théorique de $\sigma_{\mathbb{T}}(D_j)$ est donnée par :

$$\forall test_i \in \mathbb{T}, \begin{cases} (\sigma_{\mathbb{T}}(D_j))_i = 0, & \text{si } Modes(D_j) \cap Expl(test_i) = \emptyset \\ (\sigma_{\mathbb{T}}(D_j))_i = 1, & \text{si } Modes(D_j) \cap Expl(test_i) \neq \emptyset \end{cases} \tag{.0.13}$$

La distance entre les deux signatures s'écrit alors :

$$distance_{\mathbb{T}}(D_j) = \frac{\| \sigma^*_{\mathbb{T}} - \sigma_{\mathbb{T}}(D_j)) \|_1}{card(\mathbb{T})} \tag{.0.14}$$

où :

- $(\sigma^*_{\mathbb{T}})_i = \mu(test_i)$
- $\sigma_{\mathbb{T}}(D_j))$ est évalué comme indiqué précédemment
- une norme 1 [4] a été choisie pour resté cohérent avec la distance de Hamming de la logique nette de l'équation (.0.2).

Exemple

Prenons l'exemple d'un système de 2 cuves décrit par les modèles élémentaires suivants

$$\left(ok(bac1), S\frac{dh_1}{dt} - q_{e_1} + kh_1 = 0, 0 \leq h_1 < 50\text{cm}\right)$$
$$(ok(bac1), q_{s_1} - kh_1 = 0)$$
$$\left(ok(bac2), S\frac{dh_2}{dt} - q_{e_2} - kh_1 + kh_2 = 0, 0 \leq h_2 < 50\text{cm}\right)$$
$$(ok(bac2), q_{s_2} - kh_2 = 0)$$
$$(ok(capteur1), obs(t, h_1) - h_1 = 0)$$
$$(ok(capteur2), obs(t, h_2) - h_2 = 0)$$
$$(ok(vanne1), obs(t, q_{e_1}) - q_{e_1} = 0)$$
$$(ok(vanne2), obs(t, q_{e_2}) - q_{e_2} = 0)$$

En partant de ces modèles élémentaires, nous avons déduit trois sous-systèmes testables résumés par la table de signature suivante :

[3] s'il y a plusieurs jeux d'observations, $\sigma^*_{\mathbb{T}}$ correspond au maximum de $1 - \Gamma(\mu_{\mathscr{K}_{test_i}}, \mu_{\mathscr{K}'_{test_i}})$ sur tous les jeux

[4] somme des valeurs absolues

	$ok(bac1)$	$ok(bac2)$	$ok(capteur1)$	$ok(capteur2)$	$ok(vanne1)$	$ok(vanne2)$
t_1	1	1	0	1	1	1
t_2	1	1	1	1	0	1
t_3	1	0	1	0	1	0

Trois générateurs de résidus à base d'observateurs d'état ont alors été conçus. Le premier générateur, que nous appellerons $test_1$, est décrit par :

$$\frac{d}{dt}\begin{pmatrix}\hat{h_1}\\ \hat{h_2}\end{pmatrix} = \begin{pmatrix}-\frac{k}{S} & -K_1\\ \frac{k}{S} & -\frac{k}{S}-K_2\end{pmatrix}\begin{pmatrix}\hat{h_1}\\ \hat{h_2}\end{pmatrix} + \begin{pmatrix}\frac{1}{S} & 0 & K_1\\ 0 & \frac{1}{S} & K_2\end{pmatrix}\begin{pmatrix}obs(t,q_{e_1})\\ obs(t,q_{e_2})\\ obs(t,h_2)\end{pmatrix} \tag{.0.15a}$$

$$obs(t,h_2) = \begin{pmatrix}0 & 1\end{pmatrix}\begin{pmatrix}\hat{h_1}\\ \hat{h_2}\end{pmatrix} + \varepsilon_1 \tag{.0.15b}$$

$$0 \leq \hat{h_1} \leq 50cm \tag{.0.15c}$$

$$0 \leq \hat{h_2} \leq 50cm \tag{.0.15d}$$

Les contraintes (.0.15a) et (.0.15b) permettent de tester le modèle alors que les contraintes (.0.15c) et (.0.15d) permettent de tester la validité du test. Imaginons que les symptômes observés soient :

générateur de résidus	$\mu(\mathscr{K})$	$\mu(\mathscr{K}')$
$test_1$	0.75	0.8
$test_2$	0.125	0.8
$test_3$	0.3	1

Pour chaque test, il faut calculer le degré de satisfaction $\Gamma(\mu(\mathscr{K}_{test}), \mu(\mathscr{K}'_{test}))$ et la signature effective $\sigma^*_{\mathbb{T}}$ en utilisant la formule (.0.9) :

générateur de résidus	$\Gamma(\mu(\mathscr{K}), \mu(\mathscr{K}'))$	$\sigma^*_{\mathbb{T}}$
$test_1$	0.7	0.3
$test_2$	0.2	0.8
$test_3$	0.3	0.7

Avant d'étudier les résultats obtenus pour différents scénarios de défaut, observons qu'il est inutile de distinguer des items non discriminables. Ils ne feront que démultiplier le nombre de diagnostics, aussi est-il préférable de regrouper les items non-discriminables dans un macro-item. En considérant la table de signature correspondant aux 3 tests, on s'aperçoit que les items *bac*2, *capteur*2 et *vanne*2 sont non-discriminables. Pour réduire les écritures, nous introduisons un macro-item *ensemble*2 qui réunit *bac*2, *capteur*2 et *vanne*2. La nouvelle table de signature devient celle représentée dans le tableau 10.

	$ok(bac1)$	$ok(capteur1)$	$ok(vanne1)$	$ok(ensemble2)$
t_1	1	0	1	1
t_2	1	1	0	1
t_3	1	1	1	0

TAB. 10 – Table de signatures réduite du système de cuves

En exploitant la table de signature 10, on déduit les diagnostics qui suivent :

$$cfm\ (bac1) \vee cfm\ (capteur1) \vee cfm\ (vanne1) \vee cfm\ (ensemble2)$$

Le degré de vérité de chaque diagnostic est donné par :

$$\mu(\{cfm\ (bac1)\}) = max(1-0.7, 1-0.2, 1-0.3) = 0.8$$
$$\mu(\{cfm\ (capteur1)\}) = max(1-0.2, 1-0.3) = 0.8$$
$$\mu(\{cfm\ (vanne1)\}) = max(1-0.7, 1-0.3) = 0.7$$
$$\mu(\{cfm\ (ensemble2)\}) = max(1-0.7, 1-0.2) = 0.8$$

Les signatures théoriques sont données par la table de signature (10). Il est alors possible de calculer les distances entre la signature effective et la signature théorique pour chaque diagnostic :

$$distance(\{cfm\ (bac1)\}) = 0.4$$
$$distance(\{cfm\ (capteur1)\}) = 0.27$$
$$distance(\{cfm\ (vanne1)\}) = 0.6$$
$$distance(\{cfm\ (ensemble2)\}) = 0.53$$

Grâce aux deux mesures, degré de vérité et distance, il est possible d'établir un classement des diagnostics par ordre décroissant de vraisemblance :

$$\mu(cfm\ (capteur1))$$
$$\mu(cfm\ (bac1))$$
$$\mu(cfm\ (ensemble2))$$
$$\mu(cfm\ (vanne1))$$

Considérons un autre exemple où certains tests sont totalement insatisfaits :

générateur de résidus	$\mu(\mathscr{K})$	$\mu(\mathscr{K}')$
$test_1$	0	1
$test_2$	0	1
$test_3$	0.8	1

On calcule les degrés de satisfaction :

générateur de résidus	$\Gamma(\mu(\mathscr{K}),\mu(\mathscr{K}'))$	σ^*
$test_1$	0	1
$test_2$	0	1
$test_3$	0.8	0.2

Comme il y a deux tests totalement insatisfaits, les diagnostics sont calculés uniquement à partir de $test_1$ et de $test_2$:

$$cfm\ (bac1)$$
$$cfm\ (ensemble2)$$
$$cfm\ (capteur1) \wedge cfm\ (vanne1)$$

Comme nous l'avons vu précédemment, puisqu'il existe des tests totalement insatisfaits, le degré de vérité de chaque diagnostic vaut 1.

Il ne reste plus qu'à calculer les distances :

$$distance(cfm\ (bac1)) = 0.27$$
$$distance(cfm\ (ensemble2)) = 0.07$$
$$distance(cfm\ (capteur1) \wedge cfm\ (vanne1)) = 0.27$$

Le diagnostic le plus vraisemblable est $cfm\ (ensemble2)$. Parmi les deux autres diagnostics qui ont les mêmes distances, $cfm\ (bac1)$ pourrait être considéré comme plus vraisemblable en tenant compte des probabilités de défaut a priori et en supposant tous les défauts équiprobables.

Une des principales limites de cette approche est de ne pas s'appliquer aux systèmes multi-modaux. En effet, si $Modes(item) = \{ok, leak, cfm\}$ et si le degré d'appartenance au mode ok, $\mu(ok(item))$ est connu, il n'est pas possible de déduire le degré d'appartenance des autres modes. Nous nous mettrons dans le contexte où $\mu(cfm) = 1 - \mu(ok)$, c'est-à-dire où $Modes(item) = \{ok, cfm\}$. Néanmoins, dans la pratique, cette limite n'est pas trop restrictive : tous les systèmes industriels sur lesquels nous avons travaillé ne comportaient que des modèles de bon fonctionnement. A l'instar des médecins, un processus de diagnostic commence souvent par tester un système en s'appuyant sur des références de comportement attendu plutôt que d'envisager d'emblée des modes de défauts dont la combinatoire est très grande.

Bibliographie

K. H. ADJALLAH : *Contribution au diagnostic de systèmes par observateur d'état.* Thèse de doctorat, Institut National Polytechnique de Lorraine, 1993.

A. ALLAHHAM, S. PLOIX, A. A. YASSINE et Q. H. GIAP : Fault diagnosis in a plug-and-play context. *In Conference on Control and Fault Tolerant Systems SYSTOL10*, Nice, France, 2010.

K. APT : *Principles of Constraint Programming*. Cambridge University Press, New York, NY, USA, 2003. ISBN 0521825830.

K. AUTIO : Abstraction of behaviour and structure in model-based diagnosis. *In the Sixth International Workshop on Principle of Diagnosis (DX95)*, p. 1–7, 1995.

K. AUTIO et R. REITER : Structural abstraction in model based diagnosis. *In The 13th European Conference on Artificial Intelligence (ECAI-98)*, p. 269–273. John Wiley and Sons, 1998.

M. BLANKE, M. KINNAERT, J. LUNZE et M. STAROSWIECKI : *Diagnosis and Fault-Tolerant Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 3540356525.

J. CASSAR, M. STAROSWIECKI et V. COCQUEMPOT : Optimal residual design for model-based fault detection and isolation. *In 3rd European Control Conference ECC 95*, Roma, Italy, 1995.

D. CAYRAC, D. DUBOIS et H. PRADE : Handling uncertainty with possibility theory and fuzzy sets in satellite fault diagnosis application. *IEEE transactions on Fuzzy Systems*, 4 (3):251–269, 1996.

L. CHITTARO et A. KUMAR : Reasoning about function and its applications to engineering. *Artificial Intelligence in Engineering*, 12(4):331–336, 1998.

L. CHITTARO et R. RANON : Hierarchical model-based diagnosis based on structural abstraction. *Artificial Intelligence*, 1-2:147–182, 2004.

M. O. CORDIER, P. DAGUE, M. DUMAS, F. LÉVY, J. MONTMAIN, M. STAROSWIECKI et L. TRAVÉ-MASSUYÈS : A comparative analysis of ai and control theory approaches to

model-based diagnosis. *In 14th European Conference on Artificial Intelligence*, Berlin, Germany, 2000.

M.-O. CORDIER, P. DAGUE, F. LÉVY, J. MONTMAIN, M. STAROSWIECKI et L. TRAVÉ-MASSUYÈS : Conflicts versus analytical redundancy relations, a comparative analysis of the model-based diagnosis approach from the artificial intelligence and automatic control perspectives. *IEEE Transactions on Systems, Man, and Cybernetics*, p. 2163–2177, 2004.

R. DAVIS et W. HAMSCHER : *Model-based reasoning : troubleshooting*, p. 297–346. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

R. DAVIS : Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24 (1-3):347 – 410, 1984. ISSN 0004-3702.

J. DE KLEER et B. C. WILLIAMS : Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

J. DE KLEER : How circuits work. *Artificial Intelligence*, 24(1-3):205–280, 1984. ISSN 0004-3702.

J. DE KLEER et B. C. WILLIAMS : Diagnosis with behavioral modes. *Readings in model-based diagnosis*, p. 124–130, 1992.

M. DESINDE, J.-M. FLAUS et S. PLOIX : Risk analysis and diagnosis modelling for online control of process. *In Proceedings of the Safety and Reliability ESREL2006 Conference*, 2006.

M. DESINDES : *Contribution à la mise au point d'une approche intégrée analyse diagnostique / analyse de risques*. Thèse de doctorat, Grenoble Institute of Technology (INPG), 2006.

K. DOWNING : The qualitative criticism of circulatory models via bipartite teleological analysis. *Artificial Intelligence in Medicine*, 2(3):149 – 171, 1990. ISSN 0933-3657.

D. DUSTEGOR, V. COCQUEMPOT et M. STAROSWIECKI : Structural analysis for residual generation : Towards implementation. *In the IEEE International Conference on Control Applications*, p. 1217–1222, Taipei, Taiwan, 2004.

J. M. FLAUS : Modelisation de systemes organisationnels pour l'analyse des defaillances : application au plan de sauvegarge communal. *In 8ème ENIM IFAC Conférence Internationale de Modélisation et Simulation, Tunisie*, 2010.

J. M. FLAUS et O. GRANDAMAS : Towards a formalisation of mads, system failure analysis model. *In ESREL 2002, 19-22 avril , Lyon*, 2002.

J. M. FLAUS : Un modèle de danger unifié pour l'analyse systémique des risques. *In in : Eds. Lavoisier Tec & Doc, Récents Progrès en Génie des Procédés 90, Paris*, 2003.

J. M. FLAUS : A model-based approach for systematic risk analysis. *Proceedings of the Institution of Mechanical Engineers, Part O : Journal of Risk and Reliability*, Volume 222, Number 1 / 2008:79–93, 2008.

J. M. FLAUS et H. T. PHAM : Human-computer interactive and iterative diagnosis using a mixed structuro-functionnal model. *In Internal Report, submitted to ESREL Conference*, 2010.

J. M. FLAUS, E. PIATYSZEK, G. KARAGIANNIS et J. R. JACOB : Diagnostic des dysfonctionnements des plans de secours pour la gestion des risques majeurs. *In Rapport Interne, soumis à la conférence STIC & Environnement*, 2010.

P. FRANK : Analytical and qualitative model-based fault diagnosis - a survey and some new results. *European Journal of Control*, 2:6–28, 1996.

D. W. FRANKE : Deriving and using descriptions of purpose. *IEEE Expert : Intelligent Systems and Their Applications*, 6(2):41–47, 1991. ISSN 0885-9000.

E. FRISK : Residual generator design for non-linear, polynomial systems-a grobner basis approach. *In SAFEPROCESS'2000*, Budapest, Hungary, 2000.

L. FROQUET et J. M. FLAUS : Structural modelling of automatic food processing for hazard studies. *In CESA'03, Lille*, 2003.

M. R. GENESERETH : The use of design descriptions in automated diagnosis. *Artif. Intell.*, 24 (1-3):411–436, 1984. ISSN 0004-3702.

J. GERTLER : Analytical redundancy methods in fault detection and isolation. *In International Conference on fault diagnosis Tooldiag'93*, Toulouse, France, 1993.

Q. H. GIAP, O. ADROT, S. PLOIX, C. DEPRAETERE et J. M. FLAUS : Fuzzy reasoning based interactive diagnosis of a grid network of rain gauge sensors. *In The 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, Valenciennes, France, 2010a.

Q. H. GIAP, S. PLOIX et J. M. FLAUS : Managing diagnosis processes with interative decompositions. *In Proceedings of the Artificial Intelligence Applications and Innovations III*, 296:407–415, 2009.

Q. H. GIAP, S. PLOIX et J. M. FLAUS : Cooperation homme-machine dans l'analyse diagnostique. *In Sixième Conference Internationale Francophone d'Automatique*, Nancy, France, 2010b.

Q. H. GIAP, S. PLOIX et J. M. FLAUS : Modélisation itérative structuro-fonctionnelle pour le diagnostic. *In 8ème ENIM IFAC Conférence Internationale de Modélisation et Simulation*, Tunis, Tunisie, 2010c.

R. GREINER, B. A. SMITH et R. W. WILKERSON : A correction to the algorithm in reiter's theory of diagnosis. *Artificial Intelligence*, 41(1):79–88, 1989.

J. T. H. RESSENCOURT, L. Travé-Massuyès : Hierarchical modelling and diagnosis for embedded systems. *17th International Workshop on Principles of Diagnosis DX'06, Aranda de Duero (Spain)*, p. pp. 235–242, June 26-28, 2006.

IEEE : *STD 1320.1-1998. Norme d'IEEE pour la Langue-Syntaxe et la sémantique modelantes fonctionnelles pour IDEF0*. New York : IEEE, 1998.

R. ISERMANN : Model-based fault detection and diagnosis-status and applications. *In 16th symposium on Automatic Control in Aerospace (ACA'2004)*, Petersbourg, Russia, 2004.

G.-M. KARAGIANNIS : *Méthodologie pour l'analyse de la robustesse des plans de secours industriels*. Thèse de doctorat, Docteur de l'Ecole Nationale Supérieure des Mines de Saint-Etienne Spécialité : Sciences et Génie de l'Environnement, 2010.

A. M. KEUNEKE : Device representation-the significance of functional knowledge. *IEEE Expert : Intelligent Systems and Their Applications*, 6(2):22–25, 1991. ISSN 0885-9000.

S.-H. KIM et K.-J. JANG : Designing performance analysis and idef0 for enterprise modelling in bpr. *International Journal of Production Economics*, 76(2):121 – 133, 2002. ISSN 0925-5273.

J. D. KLEER et B. C. WILLIAMS : Diagnosing multiple faults. *Artificial Intelligence*, 32 (1):97–130, 1987.

J. L. KOLODNER : *Case Based Reasoning*. Morgan Kaufmann, 1993.

I. MOZETIC : *Hierarchical model-based diagnosis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992. ISBN 1-55860-249-6.

P. P. NAYAK : Diagnosis with multiple theories. *In Proceedings of the Fifth International Workshop on Principle of Diagnosis (DX-94)*, p. 217–225, 1994.

M. T. NGUYEN : Détection et localisation de défauts sur un réseau de capteurs hydrologiques. Mémoire de D.E.A., Grenoble Institute of Technology (INPG), 2005.

M. NYBERG et E. FRISK : Residual generation for fault diagnosis of systems described by linear differential-algebraic equations. *IEEE Transactions on Automatic Control*, 51:1995–2000, 2006.

M. NYBERG et M. KRYSANDER : Combining AI, FDI, and statistical hypothesis-testing in a framework for diagnosis. *In Proceedings of IFAC Safeprocess'03*, Washington, USA, 2003.

R. PATTON, P. FRANK et R. CLARK : *Fault diagnosis in dynamic systems*. Prentice Hall, 1989.

S. PLOIX et P. CHAZOT : Iterative expert driven fault diagnosis based on structural modeling. *In IFAC Symposium SAFEPROCESS'2006*, August 2006.

S. PLOIX, M. DESINDE et S. TOUAF : Automatic design of detection tests in complex dynamic systems. *In The 16th IFAC World Congress*, Prague, Czech republic, 2005a.

S. PLOIX, M. DESINDE et S. TOUAF : Automatic design of detection tests in complex dynamic systems. *In 16th IFAC World Congress*, Prague, Czech republic, 2005b. 4-8 july.

S. PLOIX, S. TOUAF et J. M. FLAUS : A logical framework for isolation in fault diagnosis. *In SAFEPROCESS'2003*, Washington D.C., U.S.A., 2003.

S. PLOIX, A. YASSINE et J.-M. FLAUS : An improved algorithm for the design of testable subsystems. *In The 17th IFAC World Congress*, Seoul, Corea, 2008.

S. PLOIX, A. YASSINE et J.-M. FLAUS : A new efficient and flexible algorithm for the design of testable subsystems. *Applied Mathematics and Computer Science*, To appear, 2009.

S. PLOIX, A. YASSINE et J. M. FLAUS : A new efficient and flexible algorithm for the design of testable subsystems. *Applied Mathematics and Computer Science*, 20(1), 2010.

C. PRICE : *Computer-Based diagnostic systems*. Springer-Verlag, 1999.

J. R. QUINLAN : Induction of decision trees. *Machine Learning*, 1:81–106, 1986. ISSN 0885-6125.

R. REITER : A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

M. STAROSWIECKI et G. COMTET-VARGA : Analytical redundancy relations for fault detection and isolation in algebraic dynamic systems. *Automatica*, 37:687–699, 2001.

L. STEELS : Diagnosis with a function-fault model. *Appl. Artif. Intell.*, 3(2-3):213–237, 1989. ISSN 0883-9514.

J. STICKLEN et W. E. BOND : Functional reasoning and functional modelling. *IEEE Expert : Intelligent Systems and Their Applications*, 6(2):20–21, 1991. ISSN 0885-9000.

P. STRUSS : What's in sd ? towards a theory of modeling for diagnosis. *In* W. HAMSCHER, L. CONSOLE et J. DE KLEER, éds : *Readings in model-based diagnosis*, p. 419–448. Morgan Kaufman, 1992.

P. STRUSS et O. DRESSLER : "physical negation" : Integrating fault models into the general diagnostic engine. *In Proc. of the 11th IJCAI*, p. 1318–1323, Detroit, MI, 1989.

L. THEVENON et J. M. FLAUS : Modular representation of complex hybrid systems : Application to the simulation of batch processes. *Simulation Practice and Theory*, Volume 8, Issue 5:pp. 283–306, 2000.

S. TOUAF : *Diagnostic logique des systèmes complexes dynamiques dans un contexte multi-agent*. Thèse de doctorat, Grenoble Institute of Technology (INPG), 2005.

S. TOUAF et S. PLOIX : Diagnosis for large scale distributed industrial plants - application to an hydraulic looper. *In 11th IFAC Symposium on Automation in Mining, Mineral and Metal processing, MMM04*, Nancy, France, 2004a.

S. TOUAF et S. PLOIX : Soundly managing uncertain decisions in diagnostic analysis. *In 15th International Workshop on Principles of Diagnosis (DX2004)*, 2004b.

L. TRAVÉ-MASSUYÈS, T. ESCOBET et X. OLIVE : Diagnosability analysis based on component supported analytical redundancy relations. *17th International Workshop on Principles of Diagnosis DX'06, Aranda de Duero (Spain)*, p. pp. 235–242, 2006.

I. WATSON : *Applying Case-Based Reasoning : Techniques for Enterprise Systems*. Morgan Kaufmann, 1997.

XRISK : www.xrisk.fr.

R. R. YAGER : A characterization of the extension principle. *Fuzzy Sets and Systems*, 18:205–217, 1986.

A. YASSINE : *Génération des tests et placement de capteurs pour le diagnostic des systèmes physiques s'appuyant sur une modélisation structurelle*. Thèse de doctorat, Grenoble Institute of Technology (INPG), 2008.

L. A. ZADEH : Fuzzy sets. *Information and Control*, 8:338–353, 1965.

L. A. ZADEH : The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, 8:199–249, 1975.

A. ZAKARIAN et A. KUSIAK : Process analysis and reengineering. *Computers & Industrial Engineering*, 41(2):135 – 150, 2001. ISSN 0360-8352.

Résumé : Cette thèse étudie les problèmes de diagnostic itératif et propose des outils d'aide au diagnostic interactif. Différents processus de diagnostic où des outils interactifs homme-automate sont utiles, sont présentés. Ces outils permettent de résoudre des difficultés liées à la représentation d'un grand nombre d'éléments d'un système, des difficultés liées à la représentation du comportement et du fonctionnement d'un système et des difficultés liées à l'explicitation de l'expertise. Nos travaux ont conduit à la conception de différents types d'outils interactifs d'aide au diagnostic. Le premier permet d'exploiter des représentations structuro-fonctionnelles pour construire et résoudre progressivement un problème de diagnostic. Le second outil interactif permet d'exploiter des modèles de comportement construit au fur et à mesure de la résolution d'un problème de diagnostic. Enfin, un dernier outil a été proposé pour montrer qu'il est possible de prendre compte la connaissance implicite d'un expert dans la résolution de problème de diagnostic. Un problème de diagnostic est donc présenté comme un processus itératif avec des interactions homme-automate.

Mots clés : interaction homme-machine, analyse diagnostique, détection de défauts, modélisation structurelle, modélisation comportementale, modélisation fonctionnelle, théorie de l'abstraction.

Resum : This PhD thesis studies the iterative diagnosis problems and provides the computer-aided diagnostic tool for interactive diagnosis. Different diagnosis processes where the tool to support human-machine interaction are useful, are presented. These tools help to tackle difficulties related to the representation of a large number of elements in a system, difficulties related to the representation of the behavior functioning of a system and difficulties encountered while expliciting the expertise. Our work led to the design of different interactive tools to support the diagnosis process. The first tool allows to exploit the structural-functional modeling to build and solve progressively a diagnosis problem. The second interactive tool allows to exploit the behavioral models built step by step in the diagnosis process and to solve the diagnosis problem. The final tool was proposed to show that it is possible to take into account the implicit knowledge of an expert in order to solve the diagnosis problem. A diagnosis problem is therefore presented as an iterative process with human-machine interactions.

Keywords : human-machine interaction, diagnosis analysis, fault detection, structural modeling, behavioral modeling, functional modeling, theory of abstraction.