Diagnostic : étude d'un raisonnement complexe et multi-dimensionnel

Yannick Pencolé

HABILITATION À DIRIGER DES RECHERCHES

présentée devant

l'Université Toulouse III- Paul Sabatier
Membre de l'UNIVERSITE FEDERALE TOULOUSE MIDI-PYRENEES

spécialité : INFORMATIQUE

par

Yannick Pencolé

Diagnostic : étude d'un raisonnement complexe et multi-dimensionnel

soutenue le 19 septembre 2018 devant le jury composé de :

Michel Combacau, professeur des universités, Université de Toulouse III
Philippe Dague, professeur des universités, Université Paris-Sud [Président]
Antoine Grall, professeur des universités, Université Technologique de Troyes
Stefan Haar, directeur de recherche INRIA, LSV-CNRS, ENS de Cachan
Pierre Marquis, professeur des universités, Université d'Artois
Louise Travé-Massuyès, directeur de recherche CNRS, LAAS, Toulouse

au vu des rapports de :

Antoine Grall, professeur des universités, Université Technologique de Troyes
Stefan Haar, directeur de recherche INRIA, LSV-CNRS, ENS de Cachan
Pierre Marquis, professeur des universités, Université d'Artois

Remerciements

En tout premier lieu je tenais à remercier tous les membres de mon jury pour avoir accepté de participer à cette habilitation. Merci donc à Philippe Dague mon rapporteur de thèse de doctorat qui a été promu président de mon jury d'habilitation avec qui j'ai eu un énorme plaisir à échanger au cours des différents projets qu'on a eu ensemble. Merci à mes trois rapporteurs, je leur ai imposé la lecture d'un gros volume certes, mais je l'ai écrit avec plaisir et j'espère qu'ils l'ont lu avec plaisir. Merci donc à Pierre Marquis pour cette tâche mais également pour avoir accepté de rapporter la thèse de Nuno Bélard. Merci à Stefan Haar, qui, depuis notre rencontre à l'IRISA, a toujours eu une influence sur ma recherche en diagnosticabilité et réseau de Petri, merci à lui également pour avoir rapporté la thèse de Houssam-Eddine Gougam. Enfin, merci à Antoine Grall d'avoir accepté le jeu de l'Université Paul Sabatier et d'être devenu mon rapporteur mystère. Merci à Louise, notre cheffe d'équipe éternelle et ma tutrice d'habilitation, pour toutes les opportunités et pour tous les projets auxquels elle m'a proposé de participer. Et enfin, merci à Michel pour m'avoir accepté dans son bureau et dans sa science et pour m'avoir donné l'expérience de deux encadrements de thèses enrichissantes, (le bel échange d'un 27 et d'un 61 !).

Et bien sûr, il y a les autres et en particulier mes anciens doctorants : Anika, Pauline, Nuno, Houssam, Xingyu. Ils m'ont offert leur confiance, ce qui a été essentiel dans la démarche scientifique à laquelle nous nous sommes tous attelés. Je pense avoir eu la chance de rencontrer des étudiants de cultures différentes dans des lieux différents ce qui a conduit à des échanges riches autant sur le plan scientifique que sur le plan humain. La science doit se nourrir de tous les regards, de tous les points de vue, c'est le seul moyen de ne pas passer à côté d'une idée novatrice, de casser les formats de pensées et leur carcans. Merci également pour leur sens de l'humour au quotidien et leur politesse pour avoir accepté le mien. Merci à tous les membres du RSISE d'avoir accepté un *Frenchy* et à tous les membres de DISCO d'avoir accepté un informaticien. Ce travail est l'aboutissement des nombreux échanges et collaborations que j'ai pu avoir avec vous tous. Je voudrais remercier plus particulièrement Audine Subias et Sylvie Thiébaux qui m'ont également donné l'opportunité de rencontrer et de co-encadrer Houssam et Anika.

Enfin, je voudrais remercier celle qui ne veut pas être remerciée (je le fais en dos de page, cela ne se verra pas…). Que cela soit en sortant du RSISE ou du LAAS, elle a toujours été là pour me soutenir et me faire découvrir tant de choses sur la nature humaine à travers ce monde.

Table des matières

Préambule

Ce document est un rapport d'habilitation à diriger des recherches. Ce document a ainsi pour but de donner à qui de droit les moyens de statuer sur mon aptitude à l'obtention d'une habilitation à diriger des recherches. De quoi parlons-nous donc ? Faisant référence au texte légal initial de l'Arrêté du 23 novembre 1988 relatif à l'habilitation à diriger des recherches, la réponse à cette question se trouve dans le premier paragraphe de son article 1 que je cite ici dans son intégralité.

> L'habilitation à diriger des recherches sanctionne la reconnaissance du haut niveau scientifique du candidat, du caractère original de sa démarche dans un domaine de la science, de son aptitude à maîtriser une stratégie de recherche dans un domaine scientifique ou technologique suffisamment large et de sa capacité à encadrer de jeunes chercheurs.

On distingue ainsi quatre critères à évaluer et je me propose à travers ce document de fournir les éléments nécessaires à cette évaluation.

1. *La reconnaissance du haut niveau scientifique du candidat.* Dans ce document, je fournis une synthèse de mes travaux postdoctoraux qui ont été publiés et validés par mes pairs ainsi qu'une description de l'ensemble des productions logicielles qui ont conduit à la réalisation de ces travaux mais également aux résultats de contrats de collaborations scientifiques tant avec des partenaires académiques qu'avec des partenaires industriels.
2. *Le caractère original de sa démarche dans un domaine de la science.* Le premier chapitre de ce document a pour objectif de préciser le champ scientifique de ma démarche globale et le dernier en précisera les perspectives. Chaque contribution sera expliquée à partir de son contexte initial.
3. *Son aptitude à maîtriser une stratégie de recherche dans un domaine scientifique ou technologique suffisamment large.* Le premier chapitre est consacré à la démarche globale de mes travaux. Les chapitres suivants ont pour objectif de situer les différents travaux effectués dans cette démarche globale. Chaque chapitre regroupe les travaux d'une thématique particulière et ne suivent donc en aucun cas un ordre chronologique, chaque thématique étant en complément d'une

autre. Le chapitre final propose également une synthèse de mes activités en adoptant un regard complémentaire.

4. *Sa capacité à encadrer de jeunes chercheurs.* Ce document et les résultats qu'il contient sont issus de différentes collaborations avec des chercheurs, enseignants-chercheurs, doctorants et stagiaires. Tout au long de cette synthèse, les collaborations et encadrements afférents à toute contribution seront systématiquement précisés.

Cessons maintenant de limiter l'habilitation à diriger des recherches à un simple paragraphe de droit afin de voir si l'habilitation n'est pas finalement la marque de l'acquisition d'une compétence nouvelle par rapport à un doctorat. J'avais conclu ma thèse en disant que son aboutissement amenait finalement à la connaissance de soi : et l'helléniste que je suis écrivait ainsi dans son préambule la fameuse phrase écrite sur le temple de Delphes et illustrant ce constat : Γνῶτι σεαυτον. Je voulais rebondir sur ce constat de jeune docteur avec les années et l'expérience acquise et ainsi formuler ce que je crois être la compétence nouvellement acquise pendant ces années post-doctorales. Et donc, l'helléniste que je suis toujours a voulu répondre sous forme de clin d'œil au jeune docteur que j'ai été. Selon moi, après la connaissance de soi, la compétence nouvelle qui donne sens à l'habilitation est la *connaissance d'autrui* : Γνῶτι αλλον !

Chapitre 1

Diagnostic : un raisonnement complexe et multi-dimensionnel

1.1 Introduction générale

Démarrons en posant une première définition conceptuelle de la notion fondamentale qui sera traitée tout au long de ce document : le diagnostic.

Définition 1.1 (Diagnostic) *Le diagnostic est un raisonnement, une démarche intellectuelle, qui consiste à identifier la nature d'une situation, d'un mal, d'une difficulté, d'un dysfonctionnement par l'interprétation de signes extérieurs.*

Avant de rentrer dans plus de détails, arrêtons-nous un instant sur cette définition. La première chose à noter est que le diagnostic est un *raisonnement*, typiquement une activité humaine. Son étude et sa simulation à l'aide d'ordinateurs est une branche de l'intelligence artificielle. L'objectif de ce raisonnement est d'*identifier la nature d'une situation*, autrement dit le diagnostic nécessite une classification *a priori* des situations (le sain et le malade, le bon fonctionnement et le dysfonctionnement, le bien et le mal, etc) et décide pour une situation donnée quelles sont les classes possibles de cette situation. Le troisième point à noter est que ce raisonnement se fait à l'aide d'une *interprétation de signes extérieurs*. Autrement dit, une sous-activité essentielle au diagnostic est la perception de signes qui sont émis et dont l'interprétation est orientée en vue de décider de la nature de la situation.

Cette première définition est très conceptuelle mais elle montre que l'activité de diagnostic est une activité très générale qui peut être utilisée dans tous les domaines et peut profiter à tous. En premier lieu, on pense bien évidemment au diagnostic médical (identifier une maladie à l'aide de symptômes) ou encore au diagnostic de pannes chez le garagiste. Mais ce raisonnement peut s'appliquer dans biens d'autres domaines socio-économiques.

Si l'on doit résumer en une phrase l'objectif des travaux présentés dans ce document, alors le but ultime est *la synthèse informatique d'un agent ou d'un ensemble d'agents ayant des capacités cognitives suffisantes pour surveiller, détecter et diagnostiquer des situations dans leur environnement*. Par agent, nous entendons ici une entité capable d'interagir avec son environnement (voir la figure 1.1) et en ce sens nous adoptons le même point de vue qu'en planification automatique [Ghallab et al., 2016]. Néanmoins, à la différence d'un problème de planification où l'agent a pour objectif d'agir sur son environnement pour le mener dans un état but, l'agent diagnostiqueur a pour objectif d'analyser son environnement en l'observant (action non-intrusive) afin qu'il affine son *état de croyance* et émette des hypothèses pouvant expliquer une situation qualifiée d'anormale dans l'environnement. En ce sens, un agent diagnostiqueur n'a pas l'objectif de modifier son environnement par une décision (même s'il peut en avoir la capacité comme c'est le cas dans le cadre d'un diagnostic actif (voir notamment la section 5.2 page 161)) mais de le comprendre par la simple observation de celui-ci.

La réalisation d'un diagnostic passe en général par un ensemble d'étapes illustrées par la figure 1.1. À partir des observations disponibles, la première d'entre elle est la *détection*. Cette étape détermine si le comportement observé est considéré comme normal ou non. Ensuite vient la phase de *localisation* : elle consiste à isoler dans l'environnement des parties suspectes qui seraient à l'origine du comportement anormal. La phase d'*identification*, quant à elle, a pour objectif de déterminer dans ces parties suspectes la nature du problème. La dernière phase, celle d'*explication*, génère la relation cause-effet entre les problèmes/dysfonctionnements identifiés et les observations, en cela elle explique la raison de telle ou telle observation (observation symptomatique). La mise en œuvre effective de ces différentes étapes dépend notamment de la disponibilité des connaissances sur le système. La détection requiert une notion de modèle fonctionnel de l'environnement (fonctionnement normal). La localisation nécessite en plus une notion de modèle structurel (un ensemble de parties). L'identification nécessite un modèle de comportement/fonctionnement en cas de défaillance et l'étape d'explication requiert en plus une information cause-effet entre le dysfonctionnement et ses symptômes. Toute étape de diagnostic peut être rendue *active*. Le diagnostic actif nécessite des moyens d'actions/planification sur l'environnement en particulier des choix de mesures, des applications de tests afin d'infirmer/confirmer des hypothèses de diagnostic.

Enfin, le diagnostic peut s'effectuer de façon collaborative par plusieurs agents diagnostiqueurs. En fonction de l'environnement à surveiller, différentes architectures d'agents peuvent être envisagées. L'*architecture centralisée* consiste en un seul agent qui a une connaissance globale de l'environnement et dispose de toutes les observations disponibles. Il fournit un diagnostic global. Dans une *architecture décentralisée/distribuée*, plusieurs agents ont une connaissance locale de l'environnement et fournissent des diagnostics locaux. Afin de garantir une cohérence globale, ils

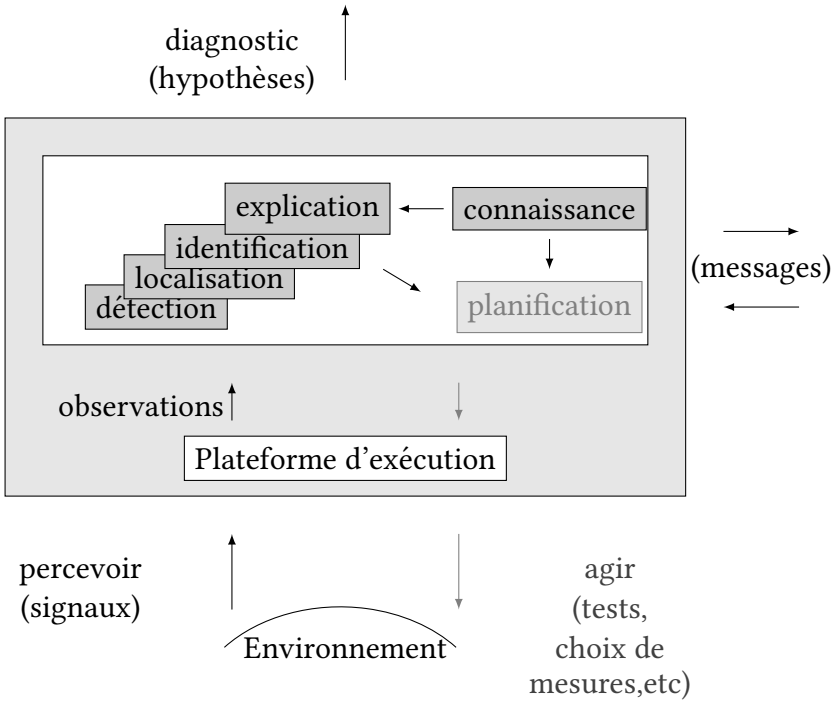


FIGURE 1.1 – Agent diagnostiqueur générique.

procèdent par échange de messages soit pour fournir un diagnostic global (dans une architecture décentralisée) soit pour fournir des diagnostics locaux mais *globalement cohérents* (architecture distribuée).

Comparer d'entrée de jeu diagnostic et planification n'est pas anodin. L'agent diagnostiqueur peut être vu entre autres comme l'assistant d'un agent planificateur. En fournissant une analyse complète de la situation environnementale au planificateur, il offre à celui-ci un moyen de revoir ses objectifs et d'affiner ses décisions d'actions pour mieux conduire son environnement dans un état but encore acceptable et d'en expliquer l'échec s'il n'est plus possible de mener l'environnement dans un tel état. Mais le diagnostic n'est pas uniquement lié au problème de planification automatique, ce raisonnement peut en effet être utilisé dans d'autres contextes en particulier ceux qui seront développés au chapitre 5.

Dans ce document, nous considérons que l'agent diagnostiqueur se focalise sur la supervision et le diagnostic d'un système dans l'environnement. Un système ici est à prendre au sens large du terme.

Définition 1.2 (Système) *Un système est un ensemble identifié d'éléments réels de l'environnement interagissant entre eux selon certaines règles et dont l'interaction avec l'environnement externe a une finalité.*

Là encore, arrêtons-nous un instant sur cette deuxième définition. Premièrement, un système est un *ensemble d'éléments réels*, autrement dit un système est une partie de la réalité. Dans ce document, nous ferons toujours la distinction entre le système et toute représentation intellectuelle de celui-ci (communément appelé modèle). Le deuxième point important ici est que cet ensemble est *identifié* comme un tout par celui qui perçoit le système (en particulier le diagnosticien) au milieu de son *environnement*. Un système interagit avec son environnement et peut, en particulier, fournir des signes extérieurs au diagnosticien. On demande également au système d'avoir une finalité. L'interaction du système avec son environnement a un but, un objectif qui peut ne pas être atteint dès lors que le système est en dysfonctionnement. Ainsi, le diagnosticien impose au système d'avoir une fonction vis-à-vis de son environnement. La vue systémique est propre au domaine de recherche en automatique. C'est par ce point de vue que toute une communauté scientifique issue de l'automatique aborde également l'étude du diagnostic de systèmes.

Par modèle de systèmes, nous adopterons le point de vue défini dans [Chittaro et al., 1993]. Un système, une réalité physique, peut être modélisé avec de nombreux points de vue (modèles fonctionnels, modèles structurels, modèles comportementaux) qui peuvent également être exploités ensemble pour obtenir un résultat plus précis ou de façon plus efficace.

Définition 1.3 (Modèle de système) *Un modèle est une représentation mathématique d'un point de vue sur le système.*

Dans les travaux présentés, différents formalismes (langages) de représentation de la connaissance sur un système seront exploités. Du point de vue du diagnostic, un modèle (peu importe sa nature) constitue ainsi la base de connaissances sur laquelle va se fonder le raisonnement.

1.2 Formalisation générique du problème de diagnostic

Soit $\mathcal{L}$ un langage de représentation de la connaissance. À l'aide de ce langage $\mathcal{L}$, on peut décrire des phrases logiques $\pi_1, \pi_2, \ldots$ et ainsi représenter dans une base de connaissance DS la *description du système* que l'on souhaite diagnostiquer par un ensemble de phrases de $\mathcal{L}$. Soit $\models$ l'opérateur sémantique de conséquence logique associé à ce langage $\mathcal{L}$, tout au long de ce document on considérera que cet opérateur est *Tarskien*, à savoir :

1. $\models$ est une relation monotone : pour toute base de connaissance B si $B \models \pi$ alors pour toute phrase π', $B \cup \{\pi'\} \models \pi$ (l'ajout de connaissance n'invalide pas de conséquence).

2. $\models$ est idempotente : si $B \models \pi$ et $B \cup \{\pi\} \models \pi'$ alors $B \models \pi'$ (l'ajout de conséquences de la base B dans B n'entraîne pas de nouvelles conséquences).
3. $\models$ est extensible : $B \models \pi$ pour tout $\pi \in B$ (toute phrase de la base B est conséquence de la base B).

Tout au long de ce document, on supposera que l'agent diagnostiqueur dispose à un instant donné d'une description du système DS (un modèle du système à diagnostiquer) d'un ensemble d'observations OBS et d'un ensemble de propriétés d'intérêts $\pi_1, \ldots, \pi_n$ et son objectif sera de *déterminer* à partir de DS et de OBS si l'une ou plusieurs de ces propriétés π_i sont possibles. Formellement, si l'on distingue dans OBS les observations symptomatiques $\mathsf{OBS_S}$ des autres $\mathsf{OBS_C}$ alors l'agent diagnostiqueur a pour objectif de déterminer les π_i telles que :

$$\mathsf{DS} \cup \{\pi_i\} \cup \mathsf{OBS_C} \cup \mathsf{OBS_S} \nvDash \bot$$

et

$$\mathsf{DS} \cup \{\pi_i\} \cup \mathsf{OBS_C} \models \mathsf{OBS_S}.$$

La propriété π_i est ainsi un diagnostic candidat dès lors que les observations sont en cohérence avec le système décrit par DS *affecté* par la propriété π_i (ce que l'on nomme diagnostic fondé sur la cohérence (*consistency-based diagnosis*)). De plus, si des observations symptomatiques $\mathsf{OBS_S}$ sont présentes alors la présence de la propriété π_i sur le système DS est une *cause possible* de ces symptômes (diagnostic abductif).

La formalisation du problème présentée ci-dessus est générique, elle repose sur un langage de représentation $\mathcal{L}$ et un opérateur $\models$ qui lui est associé. Les travaux présentés dans ce document décrivent des instances de ce problème générique où le choix de $\mathcal{L}$ est déterminé par le type d'information (la/les dimension(s)) traité sur un système donné. Nous exposerons notamment différents problèmes associés à ce problème de diagnostic. Le premier d'entre eux est bien évidemment calculatoire, le diagnostic par définition est lié à la notion de satisfaisabilité qui est déjà un problème NP-complet sur la logique propositionnelle, ce qui revient à dire que le succès d'une approche de diagnostic repose essentiellement sur l'*efficacité de la méthode proposée*. La deuxième source de difficulté est liée à la disponibilité et la qualité des observations OBS. Une trop faible disponibilité peut induire un nombre considérable de diagnostics candidats et conduire ainsi à l'échec d'un tel raisonnement (que dire en effet si le raisonnement de diagnostic aboutit à la conclusion que tout est possible ?). L'étude *a priori* du lien entre la disponibilité des observations et les potentielles ambiguïtés qui peuvent être levées grâce à elles dans DS est ce que l'on désigne par l'*étude de diagnosticabilité*, des travaux seront également exposés sur cette thématique. Enfin, une dernière difficulté dans le raisonnement de diagnostic est la qualité de la connaissance sur le système. Plus généralement le raisonnement fait certaines hypothèses sur DS, notamment que la connaissance est correcte (tout ce qui est décrit, est) voire complète (tout ce qui n'est

pas décrit, n'est pas : hypothèse du monde clos). Qu'en est-il si ces hypothèses ne sont pas effectives ? L'analyse de ces questions sont des problèmes de *méta-diagnostic*.

1.3 Organisation de ce document

La suite de ce document est organisée de la façon suivante. Les chapitres 2 à 4 présentent les différentes contributions au problème de diagnostic (diagnostic, diagnosticabilité, …). Le chapitre $n+1$, $n \in \{2,3\}$ traite du diagnostic de systèmes en prenant en compte une dimension supplémentaire par rapport aux systèmes traités dans le chapitre n. Ce classement n'est donc pas chronologique mais thématique. Avec le chapitre 2, on démarre donc avec les systèmes statiques où le temps ne modifie donc pas l'état de santé du système. Le chapitre 3 présente les travaux sur les systèmes à événements discrets, où l'état du système change avec l'occurrence d'événements au cours du temps. Le chapitre 4 introduit la dimension temporelle dans le système à savoir que la date de l'occurrence des événements a une importance contrairement aux systèmes présentés dans le chapitre 3. Le chapitre 5, quant à lui, présentera des contributions sur l'intégration du diagnostic pour l'aide à la décision (réparation automatique, autonomie décisionnelle, maintenance correctionnelle et prévisionnelle). Le chapitre 6 a pour objectif de rappeler quelques aspects autour des logiciels qui ont été développés et ont contribué à la production des résultats expérimentaux. Enfin le chapitre 7 présente les conclusions de ces travaux et les perspectives de recherche.

Chapitre 2

Diagnostic et sa vision statique

Ce chapitre est consacré au raisonnement diagnostique sur des systèmes en adoptant un point de vue purement statique. La théorie décrite ici repose sur le fait que le système étudié est considéré être dans un état de santé permanent (état anormal ou non). De même, la notion d'observation introduite ici représente une information partielle de cet état. Cette théorie constitue également la trame fondamentale des travaux qui ont été initiés dans les années 80-90 et qui ont fait émerger la communauté scientifique dite du « diagnostic à base de modèle » dont l'acronyme est DX [Hamscher et al., 1992]. Beaucoup de travaux ont déjà eu lieu sur cette thématique et cet aspect du raisonnement diagnostique a finalement été laissé de côté pendant de nombreuses années. L'un des freins majeurs dans ce domaine est la complexité des algorithmes qui nécessitent, comme nous le verrons, l'utilisation intensive de prouveurs de théorèmes logiques, d'outils de raisonnement automatiques [Mozetic and Holzbaur, 1994]. Ce n'est que très récemment, que l'on perçoit au sein de la communauté un regain d'intérêt pour ce domaine dont l'origine est la montée en puissance des ordinateurs et des progrès scientifiques notamment sur la programmation par contraintes et le calcul de satisfaisabilité de formules propositionnelles [Feldman et al., 2010a], [Feldman et al., 2014], [Marques-Silva et al., 2015], [Jannach et al., 2016], [Stern et al., 2017].

2.1 Rappels sur la théorie logique du diagnostic

Le principe fondamental de la théorie proposée par [Reiter, 1987] a été initiée par [deKleer, 1976] suivie de [Davis, 1984], [Genesereth, 1984] pour une application dans les circuits électroniques. Dans cette théorie du raisonnement de diagnostic, on distingue la connaissance du système analysé et le raisonnement lui-même qui est décrit à l'aide d'un moteur générique de diagnostic. La connaissance du système est représentée à l'aide d'un modèle décrit dans une logique. Cette

connaissance est dite *de principes premiers*, à savoir que le modèle décrit comment le système fonctionne (en particulier comment le système fonctionne normalement). Cette connaissance est généralement issue d'une modélisation qualitative du fonctionnement physique du système analysé. Cette connaissance est à opposer à la connaissance dite *de surface*, où l'on associe directement des observations (symptômes) à des causes de dysfonctionnement (connaissance utilisée en général par des systèmes experts [Giarratano and Riley, 2004]).

2.1.1 Connaissance du système et sa représentation

La connaissance du système à diagnostiquer est représentée à l'aide d'un modèle formel qui s'appuie dans le cas des systèmes logiques sur la logique du premier ordre.

Définition 2.1 (Modèle du système) *Le modèle d'un système est un couple* $(\mathsf{DS}, \mathsf{COMPS})$ *où :*

- DS *est un ensemble fini de phrases logiques du premier ordre, aussi appelé la* description du système *;*
- COMPS *est un ensemble de symboles de la logique sous-jacente, chaque symbole représentant un* composant d'intérêt *dans le système.*

La définition 2.1, issue des travaux originaux de [Reiter, 1987] et de [deKleer and Williams, 1987] est très abstraite et ne met pas véritablement en avant le fait que DS modélise le comportement d'un système. Dans la vision statique, un système réalise une fonction qui, à partir d'un ensemble d'entrées, produit une sortie. Dans [5], on se propose d'affiner cette définition (sans en restreindre son application) afin de mieux rendre compte que DS modélise le comportement d'un système.

Définition 2.2 (Paramètre [5]) *Un paramètre est un terme logique* p *désignant toute quantité physique qui peut être échangée entre le système et son environnement, le terme* $\mathsf{v}(\mathsf{p})$ *désignant sa valeur.*

Le paramètre désigne donc une quantité échangée, par exemple : une tension $\mathsf{v}(\mathsf{T}) = 10.5$, une pression $\mathsf{v}(\mathsf{p}) = [2, 5]$, un signal booléen $\mathsf{EstVrai}(\mathsf{v}(\mathsf{s}))$, … C'est à partir de ces paramètres que la fonction du système est mise en œuvre. Un paramètre peut être un paramètre d'entrée (le système n'influe pas sur sa valeur) ou un paramètre de sortie (le système influe/génère sa valeur). C'est la notion de contexte.

Définition 2.3 (Contexte [5]) *Le contexte* CXT *d'un système est une partition* $P_E \cup P_S$ *de l'ensemble de paramètres du système :* P_E *est l'ensemble des paramètres d'entrée,* P_S *est l'ensemble des paramètres de sortie. En logique un contexte est décrit par une conjonction :*

$$\mathsf{CXT} \stackrel{\text{def}}{=} \bigwedge_{\mathsf{p} \in P_E} \mathsf{Entree}(\mathsf{p}) \wedge \bigwedge_{\mathsf{p} \in P_S} \neg \mathsf{Entree}(\mathsf{p}).$$

La notion de contexte permet d'exprimer le fait qu'un système peut avoir différents modes de fonctionnement où les flux d'échanges avec l'extérieur sont différents. Quand le système n'a qu'un seul contexte, il est en général omis dans DS. Cette notion de contexte interviendra en particulier dans la théorie du méta-diagnostic (section 2.4 page 38).

Dans la description du système, on doit pouvoir ensuite distinguer les comportements dits normaux des comportements dits anormaux. C'est l'objectif du prédicat d'anomalie.

Définition 2.4 (Prédicat d'anomalie) *Le prédicat* An(.) *est le prédicat unaire qui signifie pour tout composant* c ∈ COMPS*, que le composant* c *a un comportement anormal.*

La figure 2.2 illustre une telle description logique du comportement normal du circuit logique de la figure 2.1, circuit qui a été initialement décrit dans [Davis, 1984]. L'ensemble des composants de ce circuit est :

$$\mathsf{COMPS} = \{\mathsf{M}_1, \mathsf{M}_2, \mathsf{M}_3, \mathsf{A}_1, \mathsf{A}_2\}.$$

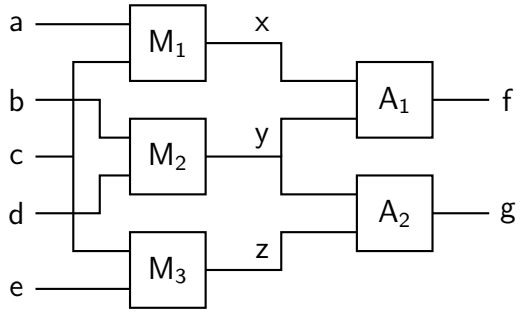


FIGURE 2.1 – Circuit logique contenant 3 multiplicateurs (M_1, M_2 and M_3) et 2 additionneurs (A_1 et A_2).

La première phrase logique décrit le contexte unique du système et chaque autre phrase logique de la figure 2.2 décrit un comportement nominal du circuit. Par exemple, la phrase M_1desc exprime que si le composant M_1 est dans un état normal alors la valeur de sa sortie est le résultat de la multiplication de ses deux entrées a et c.

Dans l'exemple précédent, on ne décrit que le comportement normal : toute phrase est de la forme ¬An(C) ⟹ ..., autrement dit : *si le composant C fonctionne normalement alors,...*. On peut évidemment intégrer à cette connaissance, des comportements anormaux : un comportement anormal est un comportement explicite et anticipé du système dans un contexte particulier (que l'on appelle couramment un mode anormal). Un mode anormal peut juste expliciter le comportement anormal d'un composant avec une phrase de la forme An(C) ⟹ ..., par exemple, si l'on sait qu'un multiplicateur

$$
\begin{array}{ll}
\mathsf{CXT}\ : & \mathsf{Entree}(a) \wedge \mathsf{Entree}(b) \wedge \mathsf{Entree}(c) \wedge \mathsf{Entree}(d) \wedge \\
& \mathsf{Entree}(e) \wedge \neg\mathsf{Entree}(f) \wedge \neg\mathsf{Entree}(g) \\
\mathsf{M_1desc}\ : & \neg\mathsf{An}(\mathsf{M_1}) \Longrightarrow (\mathsf{v(x)} = \mathsf{v(a)} * \mathsf{v(c)}) \\
\mathsf{M_2desc}\ : & \neg\mathsf{An}(\mathsf{M_2}) \Longrightarrow (\mathsf{v(y)} = \mathsf{v(b)} * \mathsf{v(d)}) \\
\mathsf{M_3desc}\ : & \neg\mathsf{An}(\mathsf{M_3}) \Longrightarrow (\mathsf{v(z)} = \mathsf{v(c)} * \mathsf{v(e)}) \\
\mathsf{A_1desc}\ : & \neg\mathsf{An}(\mathsf{A_1}) \Longrightarrow (\mathsf{v(f)} = \mathsf{v(x)} + \mathsf{v(y)}) \\
\mathsf{A_2desc}\ : & \neg\mathsf{An}(\mathsf{A_2}) \Longrightarrow (\mathsf{v(g)} = \mathsf{v(y)} + \mathsf{v(z)})
\end{array}
$$

FIGURE 2.2 – Description du comportement normal du circuit de la figure 2.1.

anormal de la figure 2.1 produit toujours une sortie à 0, on pourra ajouter au modèle, la connaissance suivante :

$$\mathsf{An}(\mathsf{M_1}) \Longrightarrow (\mathsf{v(x)} = 0) \wedge \mathsf{An}(\mathsf{M_2}) \Longrightarrow (\mathsf{v(y)} = 0) \wedge \mathsf{An}(\mathsf{M_3}) \Longrightarrow (\mathsf{v(z)} = 0).$$

On peut également intégrer des modes anormaux plus complexes intégrant plusieurs composants et des contextes comme par exemple :

$$(\mathsf{v(a)} > 50) \wedge \mathsf{An}(\mathsf{M_1}) \wedge \mathsf{An}(\mathsf{A_1}) \Longrightarrow (\mathsf{v(f)} = 50).$$

On peut enfin, introduire des modes de dysfonctionnement pour chaque composant en étendant par exemple le prédicat d'anomalie à des modes :

$$\mathsf{An_M}(\mathsf{M_1}, \mathsf{toujours0}) \Longrightarrow \mathsf{An}(\mathsf{M_1}), \mathsf{An_M}(\mathsf{M_1}, \mathsf{toujours0}) \Longrightarrow (\mathsf{v(x)} = 0)$$

2.1.2 Problème de diagnostic et solutions

À l'aide du prédicat d'anomalie, il est possible d'exprimer un état de santé pour le système. Chaque composant c est soit normal ($\neg$An(c)), soit anormal (An(c)).

Définition 2.5 (État de santé) *Un état de santé du système modélisé par* (DS, COMPS) *est une conjonction* $\sigma(\Delta)$ *de la forme :*

$$\sigma(\Delta) \stackrel{\text{def}}{=} \bigwedge_{\mathsf{c} \in \Delta} \mathsf{An(c)} \wedge \bigwedge_{\mathsf{c} \in \mathsf{COMPS} \setminus \Delta} \neg\mathsf{An(c)}$$

où Δ *est l'ensemble des composants anormaux du système.*

Il ne peut y avoir de raisonnement diagnostique sans la présence d'observations. Une *observation* est une information issue de la mesure du système. Dans cette théorie logique, les observations OBS du système sont représentées par une conjonction de *faits logiques* : chaque fait représente une information partielle mais observée de l'état

du système. Revenant sur l'exemple de la figure 2.1, si les entrées a, b, c, d, e et les sorties f, g sont observables alors la conjonction

$$v(a) = 1 \wedge v(b) = 2 \wedge v(c) = 3 \wedge v(d) = 4 \wedge v(e) = 5$$

$$\wedge v(f) = 11 \wedge v(g) = 23$$

constitue l'ensemble OBS des observations du système.

Dans la notion d'observation, on va également distinguer deux types d'observations :

1. les observations symptomatiques OBS_S sont un sous-ensemble d'observations qui sont considérées comme anormales (l'effet d'un comportement anormal) et dont on va chercher la cause par une *explication*,
2. les observations contextuelles OBS_C sont les autres observations disponibles : $OBS = OBS_S \wedge OBS_C$, les diagnostics proposés devront être cohérents avec OBS_C.

Le raisonnement de diagnostic a pour objectif de déterminer à partir de la connaissance disponible du système et des observations l'ensemble des états de santé qui sont possibles selon OBS et qui explique les symptômes OBS_S. Le problème du diagnostic peut donc s'introduire de la façon suivante.

Définition 2.6 (Problème de diagnostic logique) *Un problème de diagnostic logique* PD *s'exprime par un quadruplet* $PD \stackrel{def}{=} (DS, COMPS, OBS_C, OBS_S)$.

La *solution* de ce problème est l'ensemble des diagnostics candidats. Un candidat est un état de santé qui est *cohérent* avec le modèle (DS, COMPS) et l'observation OBS du système et qui explique les symptômes OBS_S.

Définition 2.7 (Diagnostic candidat) *Soit le problème* $PD = (DS, COMPS, OBS_C, OBS_S)$*, un* diagnostic candidat *(ou encore plus simplement un* candidat*) de* PD *est un état de santé* $\sigma(\Delta)$ *tel que :*

1. $DS \wedge OBS_C \wedge OBS_S \wedge \sigma(\Delta)$ *est satisfaisable ;*
2. $DS \wedge OBS_C \wedge \sigma(\Delta) \models OBS_S$.

La définition précédente de diagnostic candidat, ici formulée dans le cadre des systèmes logiques, illustre un des aspects fondamentaux du raisonnement diagnostique, à savoir la mise en cohérence du fait observé (les observations issues de la mesure du système réel) et la base de connaissance (condition 1). Un état de santé $\sigma(\Delta)$ est un candidat parce qu'il est un modèle de $DS \wedge OBS$, autrement dit, ce candidat rend possible l'observation de OBS dans le modèle DS : on retrouve ici l'un des principes fondateurs du diagnostic à base de modèles. Dans cette définition générale, on retrouve

également la notion d'explication de symptômes (condition 2), non seulement le candidat est cohérent avec les observations mais il est également la cause des symptômes, il existe des propagations causales dans DS qui induisent OBS_S.

Définition 2.8 (Diagnostic) *Soit* PD *un problème, sa solution, aussi appelée le* diagnostic*, est l'ensemble des diagnostics candidats de* PD.

La définition 2.6 propose en fait tout un spectre de problèmes de diagnostic [Console and Torasso, 1990], [Console and Torasso, 1991] qui est défini en fonction du choix des observations symptomatiques. On distingue en particulier les deux cas extrêmes.

1. Dans le cas où il n'y a aucune observation symptomatique $\mathsf{OBS}_\mathsf{S} = \top$, la condition 2 est automatiquement vérifiée dès lors que la condition 1 l'est. Ce cas spécifique et largement étudié est le *diagnostic de cohérence* où l'on cherche uniquement à retrouver les états de santé en cohérence avec les observations [Reiter, 1987][Hamscher, 1988], [deKleer and Williams, 1989],[Hamscher, 1988], [deKleer and Williams, 1989], [Coste-Marquis and Marquis, 1998].
2. Le cas où il n'y a que des symptômes $\mathsf{OBS}_\mathsf{C} = \top$ s'appelle le *diagnostic abductif*. Dans ce contexte DS représente en fait un graphe causal de défaillances et l'objectif est de déterminer les causes de toutes les observations [Poole, 1989a], [Peng and Reggia, 1990], [Koitz and Wotawa, 2015].

2.1.3 Algorithme de diagnostic de cohérence

Le diagnostic de cohérence est un cas particulier qui est de grande importance en réalité. À l'opposé du diagnostic abductif, il ne nécessite pas d'une connaissance a priori de comportements anormaux (du type $\mathsf{An}() \Longrightarrow$) pour retourner des solutions. Il est tout à fait possible de faire du diagnostic de cohérence en ne s'appuyant que sur la connaissance du comportement normal du système qui est en général bien plus aisée à obtenir (description de pourquoi et comment le système est fait).

Le diagnostic solution d'un problème est l'ensemble des diagnostics candidats (voir la définition 2.8) or, si l'on se limite au diagnostic de cohérence, avec une connaissance des seuls comportements normaux, on constate que si $\sigma(\Delta)$ est un état de santé candidat alors tout sur-ensemble $\Delta' \supset \Delta$ définit également un état de santé $\sigma(\Delta')$ candidat.

Définition 2.9 (Candidat minimal) *Un candidat* $\sigma(\Delta)$ *est minimal s'il n'existe pas d'autres candidats* $\sigma(\Delta')$ *tels que* $\Delta' \subset \Delta$.

Ainsi, la recherche de la solution au problème de diagnostic peut se *réduire* à la recherche de l'ensemble des candidats minimaux, car, par définition, tout candidat est un sur-ensemble d'au moins l'un de ces candidats minimaux.

2.2 DITO , un outil CSP pour le diagnostic de cohérence

Cette section présente les travaux effectués sur la thématique du diagnostic de cohérence dans les systèmes logiques.

2.2.1 Contexte des travaux

Tous les résultats qui ont été brièvement rappelés dans la section 2.1 sont des résultats théoriques fondamentaux issus principalement des travaux de la communauté DX dans les années 85-90. Liés à ces travaux théoriques, des outils développés par divers auteurs sont apparus sous le nom de moteur de diagnostic générique (GDE : *Generic Diagnostic Engine*), citons par exemple l'outil de [deKleer and Williams, 1987] ou encore l'outil de [Dressler and Struss, 1994].

Dans le cadre des travaux sur le méta-diagnostic de Nuno Belard (voir la section 2.4), il était indispensable de disposer d'un tel outil de diagnostic opérationnel et efficace. Il s'est avéré que les outils disponibles étaient inexploitables, soit ils étaient non maintenus, soit ils étaient trop spécialisés et donc incompatibles avec nos attentes.

L'objectif premier de ces travaux a donc été de mettre en œuvre un moteur de diagnostic logique mais en bénéficiant des toutes récentes avancées dans les domaines du calcul par contraintes et de la satisfaisabilité[Beldiceanu et al., 2007]. En effet, le principe du diagnostic étant la mise en œuvre d'une succession de tests de satisfaisabilité, l'idée a donc consisté à mettre à profit un outil préexistant pour les tests de satisfaisabilité, c'est l'essence même du logiciel DITO qui a été initié au cours de la thèse de Nuno Belard et développé pendant ces dernières années [38].

2.2.2 Codage d'un problème de diagnostic en CSP

L'objectif de DITO a été d'exploiter un moteur de résolution de contraintes en proposant un moyen automatique de traduire un problème de diagnostic en une succession de problèmes de satisfaction de contraintes [Rossi et al., 2006].

Définition 2.10 (Problème de satisfaction de contraintes) *Un* problème de satisfaction de contraintes *(abréviation anglaise* CSP*) est un triplet* (X, D, C) *où :*

- $X = \{x_1, \ldots, x_n\}$ *est un ensemble de* variables.
- $D = \{Dom(x_1), \ldots, Dom(x_n)\}$ *est un ensemble de* domaines. $Dom(x_i)$ *est le domaine de la variable* x_i.
- $C = \{c_1, \ldots, c_m\}$ *est un ensemble de contraintes. Une contrainte* c_i *est un sous-ensemble de* $Dom(x_1) \times \cdots \times Dom(x_n)$.

Une affectation $(v_1, \ldots, v_n)$ est un n-uplet de $Dom(x_1) \times \cdots \times Dom(x_n)$, chaque valeur v_i étant affectée à la variable correspondante x_i.

Définition 2.11 (Affectation cohérente) *Une affectation A est* cohérente *si*

$$A \in \bigcap_{i=1}^{m} c_i.$$

Une affectation cohérente est une solution du problème CSP, autrement dit la solution d'un CSP est l'ensemble $Sol(CSP)$ des affectations cohérentes. Un problème CSP est dit non-satisfaisable s'il n'existe pas de solution et donc aucune affectation cohérente :

$$Sol(CSP) = \emptyset.$$

Un outil de résolution CSP propose un langage de description de problèmes (X, D, C) utilisant un ensemble d'opérateurs afin de définir des expressions (arithmétiques ou booléennes) et des contraintes sur ces expressions. Indépendamment de l'outil utilisé, son langage peut exprimer à minima les opérateurs de contraintes suivant : ⟨et⟩, ⟨ou⟩, ⟨non⟩, ⟨égal⟩, ⟨diff⟩, ⟨inf⟩, ⟨sup⟩, ⟨infeg⟩, ⟨supeg⟩ ; ainsi que les opérateurs arithmétiques : ⟨plus⟩, ⟨moins⟩, ⟨mult⟩, ⟨div⟩, ⟨mod⟩, ⟨opp⟩. C'est avec ces opérateurs que Dito va traduire les problèmes à l'aide la fonction $encode$:

$$encode : \mathcal{L} \longrightarrow \texttt{contraintes}$$

où $\mathcal{L}$ est l'ensemble des phrases finies de la logique du premier ordre et `contraintes` est l'ensemble des contraintes pouvant être exprimées au sein de l'outil CSP. Toute phrase logique ℓ de $\mathcal{L}$ se traduit par la fonction $encode$ en un problème (X_ℓ, D_ℓ, C_ℓ). La phrase ℓ peut contenir des notations arithmétiques (tel est le cas pour la description du système de la figure 2.1). Chaque variable arithmétique v de ℓ est représentée par une variable $x_v \in X_\ell^{var}$ associée avec $Dom(x_v) \in D_\ell$ qui traduit le domaine de v sur x_v. Tout prédicat P(t) où t est un terme logique fermé est ainsi représenté par une variable CSP $x_{p_t} \in X_\ell^{pred}$ avec $Dom(x_{p_t}) = \{0, 1\} \in D_\ell$. Enfin, on a $X_\ell = X_\ell^{var} \oplus X_\ell^{pred}$. La fonction de traduction est alors définie récursivement comme suit.

1. $encode(\ell) = \langle \text{et} \rangle(encode(\ell_1), encode(\ell_2))$ si $\ell = \ell_1 \wedge \ell_2$ (conjonction).
2. $encode(\ell) = \langle \text{ou} \rangle(encode(\ell_1), encode(\ell_2))$ si $\ell = \ell_1 \vee \ell_2$ (disjonction).
3. $encode(\neg\ell) = \langle \text{non} \rangle(encode(\ell))$ (négation).
4. $encode(P(A, B, C, \dots)) = \langle \text{égal} \rangle(p_{ABC\dots}, 1)$ avec $p_{ABC\dots} \in X_\ell^{pred}$ (prédicat).
5. $encode(\forall y, f(y)) = \langle \text{et} \rangle(encode(f(Y_1)), encode(f(Y_2)), ...)$ avec $Y_i \in \bigcup_{x \in X_\ell^{var}} Dom(x)$ (quantification universelle).
6. $encode(\exists y, f(y)) = \langle \text{ou} \rangle(encode(f(Y_1)), encode(f(Y_2)), ...)$ with $Y_i \in \bigcup_{x \in X_\ell^{var}} Dom(x)$ (quantification existentielle).

7. $encode(t1\ op\ t2) = \langle \text{op} \rangle (encTerme(t1), encTerme(t2))$ avec $(op, \langle \text{op} \rangle) \in \{(=, \langle \text{égal} \rangle), (\neq, \langle \text{diff} \rangle), (<, \langle \text{inf} \rangle), (>, \langle \text{sup} \rangle), (\leq, \langle \text{infeg} \rangle), (\geq, \langle \text{supeg} \rangle)\}$ (contraintes arithmétiques).

La fonction $encTerme$:

$$\mathcal{T} \longrightarrow \texttt{expression}$$

traduit tout terme arithmétique de la logique en expression arithmétique de l'outil.

1. $encTerme(t1\ op\ t2) = \langle \text{op} \rangle (encTerme(t1), encTerme(t2))$ avec $(op, \langle \text{op} \rangle) \in \{(+, \langle \text{plus} \rangle), (-, \langle \text{moins} \rangle), (*, \langle \text{mult} \rangle), (/, \langle \text{div} \rangle), (\%, \langle \text{mod} \rangle)$ (opérateurs binaires).
2. $encTerme(-t) = \langle \text{opp} \rangle (encTerme(t))$ (opérateur unaire).
3. $encTerme(v) = v$ avec $v \in X_\ell^{var}$ et $Dom(v)$ (domaine fini).
4. $encTerme(C) = C$ (entier).

On voit à travers la simplicité de cette traduction automatique l'intérêt naturel d'exprimer une description de système en terme de contraintes, tous les calculs restant à effectuer étant à la charge du moteur de résolution CSP et de ses optimisations internes. L'outil DITO met en œuvre à l'aide de CSP un algorithme de diagnostic s'appuyant sur la recherche de conflits.

2.2.3 Rappel : algorithme de diagnostic guidé par les conflits

La mise en place d'un algorithme de diagnostic par la recherche de conflits est une technique permettant de mieux guider la recherche des diagnostics candidats [Reiter, 1987], [deKleer and Williams, 1987], [deKleer et al., 1992].

Définition 2.12 (Conflit) *Soit* $\Delta, \Delta' \subseteq \mathsf{COMPS}$*, un* conflit $\mathsf{cf}(\Delta, \Delta')$ *de* PD *est une disjonction de* $\mathsf{An}()$ *littéraux :*

$$\mathsf{cf}(\Delta, \Delta') \stackrel{\text{def}}{=} \bigvee_{c \in \Delta} \mathsf{An}(c) \vee \bigvee_{c \in \Delta'} \neg \mathsf{An}(c)$$

telle que :

$$\mathsf{DS} \wedge \mathsf{OBS} \models \mathsf{cf}(\Delta, \Delta').$$

En particulier, un conflit positif $\mathsf{cfp}(\Delta)$ *est un conflit tel que* $\mathsf{cfp}(\Delta) \stackrel{\text{def}}{=} \mathsf{cf}(\Delta, \emptyset)$.

En d'autres termes, un conflit est un *effet* (que l'on qualifie également de symptôme dans la littérature) du système DS lorsqu'il produit OBS. En particulier, dans le cas d'un conflit positif, un conflit peut être interprété comme l'affirmation qu'au moins un des composants du conflit est dans un état de fonctionnement anormal. Reprenant l'exemple de la figure 2.1 et l'observation suivante :

$$v(a) = 1 \wedge v(b) = 2 \wedge v(c) = 3 \wedge v(d) = 4 \wedge v(e) = 5$$
$$\wedge v(f) = 11 \wedge v(g) = 22,$$

comme $v(g) = 22$, si l'additionneur A_2 fonctionne correctement, on a nécessairement que $v(y) + v(z) = 22$. Si maintenant le multiplicateur M_2 fonctionne correctement on a nécessairement que $v(y) = v(b) * v(d) = 2 * 4 = 8$ ce qui impose que $v(z) = 14$. Finalement, si on suppose également que le multiplicateur M_3 fonctionne correctement alors on a $v(z) = v(c) * v(e) = 15 \neq 14$ ce qui est incohérent. Dans cet exemple, on a donc identifié trois composants $\{A_2, M_2, M_3\}$ qui ne peuvent pas être tous en bon état de fonctionnement : $An(A_2) \vee An(M_2) \vee An(M_3)$ est donc vrai dans toute interprétation de $DS \wedge OBS$ et constitue un conflit positif ($DS \wedge OBS \wedge \neg An(A_2) \wedge \neg An(M_2) \wedge \neg An(M_3)$ étant non satisfaisable, on a bien la déduction logique du conflit à partir du système et de son observation : $DS \wedge OBS \models An(A_2) \vee An(M_2) \vee An(M_3)$).

Définition 2.13 (Conflit minimal [Reiter, 1987],[deKleer and Williams, 1987]) *Un* conflit $cf(\Delta, \Delta')$ *de* PD *est* minimal *s'il n'existe pas* $\Delta_1 \subseteq \Delta$ *et* $\Delta_2 \subseteq \Delta'$ *avec* $\Delta_1 \cup \Delta_2 \subset \Delta \cup \Delta'$ *tels que* $cf(\Delta_1, \Delta_2)$ *est un conflit.*

Reprenant l'exemple précédent, il est clair que $An(A_2) \vee An(M_2) \vee An(M_3)$ est minimal. Il est également évident que $An(A_2) \vee An(M_1) \vee An(M_2) \vee An(M_3)$ est aussi un conflit mais ce dernier n'est pas minimal. Dans cet exemple, il existe un autre conflit minimal à savoir $An(A_1) \vee An(A_2) \vee An(M_1) \vee An(M_3)$. L'ensemble des conflits peut se représenter sous la forme d'un treillis (ensemble partiellement ordonné sur l'inclusion stricte ensembliste), le treillis associé à l'exemple décrit est représenté sur la figure 2.3, il indique qu'il existe en tout cinq conflits et parmi eux deux sont minimaux.

Soit ECM^+, l'ensemble des conflits minimaux positifs pour un problème donné, l'algorithme de calcul des candidats minimaux s'appuie sur le résultat fondamental suivant [Reiter, 1987] :

Théorème 2.1 *Un état de santé* $\sigma(\Delta)$ *est un candidat minimal de* PD *ssi :*

$$\sigma(\Delta) \models \bigwedge_{cf \in ECM^+} cf$$

et il n'existe pas de $\Delta' \subset \Delta$ *tel que* $\sigma(\Delta') \models \bigwedge_{cf \in ECM^+} cf$

La première condition du théorème conduit à ce que $\sigma(\Delta)$ soit un candidat, il explique tous les conflits minimaux et la deuxième condition impose que ce candidat soit minimal. Ainsi, le calcul des candidats minimaux par un algorithme guidé par les conflits s'effectue en deux étapes distinctes :

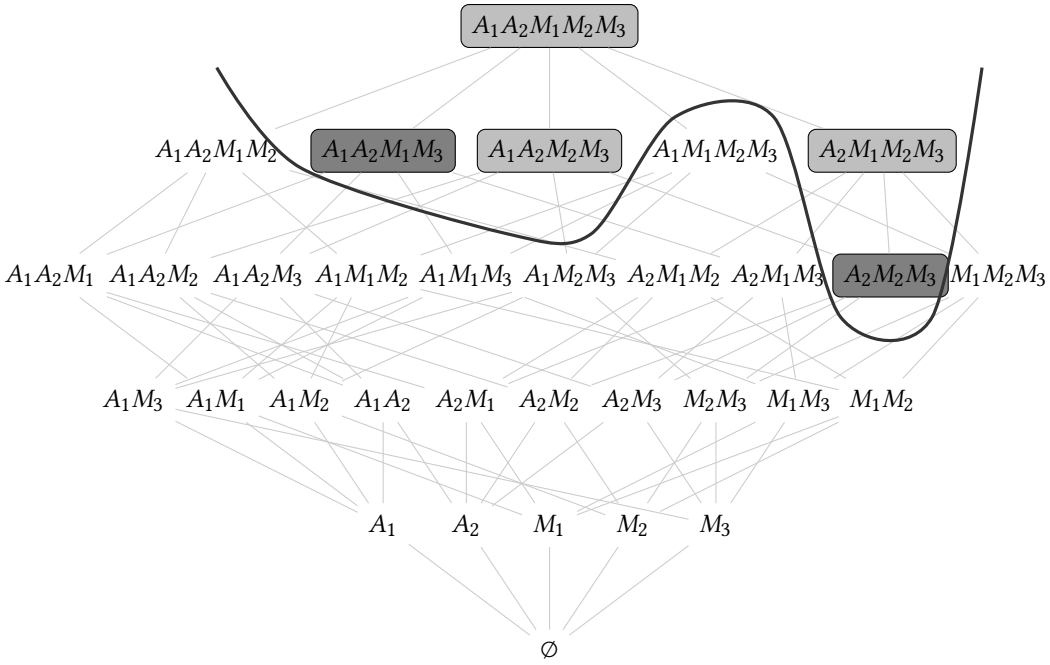


FIGURE 2.3 – Treillis des conflits de COMPS = $\{A_1, A_2, M_1, M_2, M_3\}$. Conflits minimaux : $An(A_2) \vee An(M_2) \vee An(M_3)$ et $An(A_2) \vee An(M_1) \vee An(A_1) \vee An(M_3)$.

1. Calcul des conflits minimaux ;
2. Calcul des candidats minimaux à partir des conflits minimaux.

Dans l'exemple précédent, deux conflits minimaux ont été identifiés :

$$An(A_2) \vee An(M_2) \vee An(M_3) \text{ et } An(A_1) \vee An(A_2) \vee An(M_1) \vee An(M_3).$$

À partir de ces conflits, on peut constater que $\sigma(\{A_2\})$ est un candidat minimal. En effet, il est aisé de voir que

$$\sigma(\{A_2\}) \models (An(A_2) \vee An(M_2) \vee An(M_3)) \wedge (An(A_1) \vee An(A_2) \vee An(M_1) \vee An(M_3))$$

puisque

$$\sigma(\{A_2\}) \models An(A_2).$$

De plus, il est évidemment minimal, le seul candidat plus petit étant le candidat nominal $\sigma(\emptyset)$ qui ne peut expliquer les conflits identifiés.

2.2.4 Algorithme de DITO

Considérant l'ensemble des composants COMPS, l'objectif de la recherche de conflits minimaux est de déterminer les sous-ensembles $C = \{c_{i_1}, \ldots, c_{i_k}\}$ de COMPS

tels que le $CSP(\mathcal{C})$ codant le problème $\mathsf{DS} \wedge \mathsf{OBS} \wedge (\neg\mathsf{An}(\mathsf{c}_{i_1}) \wedge \cdots \wedge \mathsf{An}(\mathsf{c}_{i_k}))$ est non-satisfaisable, autrement dit $Sol(CSP(\mathcal{C})) = \emptyset$ et pour lesquels tout sous-ensemble strict $\mathcal{C}'$ de $\mathcal{C}$ est tel que $Sol(CSP(\mathcal{C}')) \neq \emptyset$. Une façon de faire consiste à parcourir le treillis des conflits potentiels (voir la figure 2.3) en utilisant un arbre d'énumération de l'ensemble COMPS. Un arbre d'énumération permet d'explorer de façon ordonnée l'ensemble des éléments du treillis. La figure 2.4 représente le début de cet arbre dans le cas où l'ensemble des composants est celui présenté sur la figure 2.1. On parcourt les ensembles du plus petit au plus grand en imposant également un ordre pour les ensembles de mêmes tailles (ici c'est l'opposé de l'ordre lexicographique). Pour chaque nœud de l'arbre, on définit un CSP et on teste sa satisfaisabilité. Si pour un nœud donné, le CSP correspondant n'est pas satisfaisable alors le nœud représente un conflit positif.

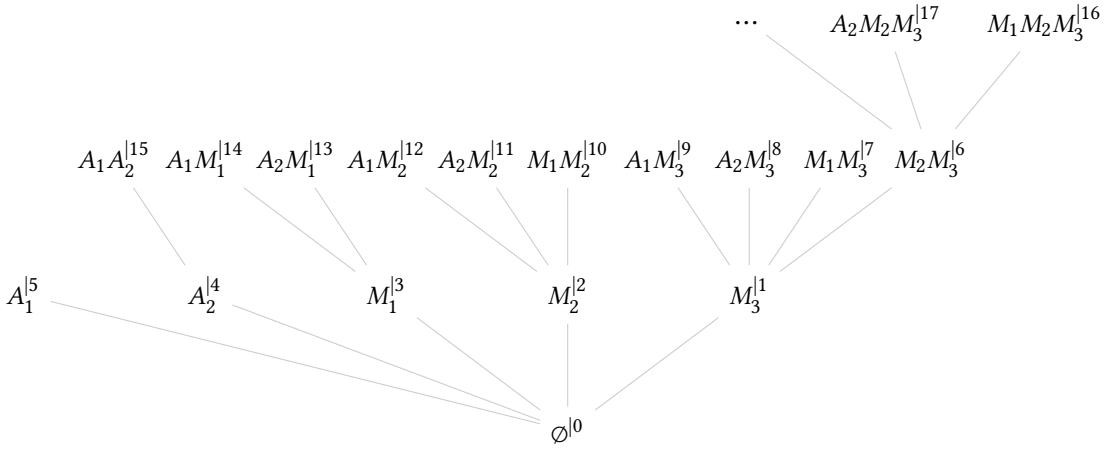


FIGURE 2.4 – Début de l'arbre énumérant l'ensemble des parties de COMPS = $\{\mathsf{A}_1, \mathsf{A}_2, \mathsf{M}_1, \mathsf{M}_2, \mathsf{M}_3\}$. L'ordre du parcours est représenté par des exposants sur chaque nœud.

L'idée est naturellement de ne pas explorer tous les nœuds de l'arbre sachant que l'on ne cherche que les nœuds ne correspondant qu'aux conflits minimaux. Cette exploration est effectuée à l'aide de l'algorithme 2.1 issu des travaux de [Zhao and Ouyang, 2006], [Zhao and Ouyang, 2007]. L'exploration de l'arbre se fait à partir du nœud racine (ligne 5) et en largeur d'abord (d'où l'utilisation d'une file $\mathcal{F}$, ligne 4). Pour chaque nœud, on retient 3 informations (C, c, E) : C est l'ensemble des composants associé au nœud lui-même, c est le composant qui a été ajouté en dernier pour arriver à ce nœud (cela identifie le nœud parent qui est $C \setminus \{c\}$) et E est l'ensemble des composants restant à intégrer à C pour visiter les nœuds enfants (ligne 9). Cette structure garantit que l'on ne parcourt un nœud qu'une fois (voir la figure 2.4). De plus, on utilise un mécanisme de propagation pour augmenter l'efficacité de la recherche : si un nœud n est associé à un conflit $\{c_{i_1}, \ldots, c_{i_k}\}$ il est inutile d'explorer tout nœud n'

associé à un ensemble $\{c_{i_1}, \ldots, c_{i_k}\} \cup \{c_{j_1}, \ldots, c_{j_l}\}$ car on sait par définition qu'il est associé lui-même à un conflit, il suffit pour cela de supprimer ces composants de la liste courante des enfants (lignes 15-18). Par construction, l'algorithme ne retient que les nœuds associés aux ensembles minimaux C pour lesquels $Sol(CSP(C)) = \emptyset$.

```
1  Fonction RechercheArbre(CSP, C);
   Entrées : CSP codage du problème
   Entrées : C ensemble de composants
2  N ← ∅;
3  si Sol(CSP) ≠ ∅ alors
4  |  F : file;
5  |  pousser((∅, ∅, C), F);
6  |  tant que F ≠ ∅ faire
7  |  |  (C, c, {c_q1, ..., c_qk}) ← oter(F);
8  |  |  pour tous les i depuis k à 1 faire
9  |  |  |  (C_i, c_qi, E_i) ← (C ∪ {c_qi}, c_qi, {c_qi+1, ..., c_qk});
10 |  |  |  CSP' ← CSP ∪ encode(¬C_i);
11 |  |  |  si Sol(CSP') = ∅ alors
12 |  |  |  |  N ← N ∪ {(C_i, c_qi, E_i)};
13 |  |  |  sinon
14 |  |  |  |  pour tous les (C_N, c_N, E_N) ∈ N faire
15 |  |  |  |  |  si c_N ∈ {c_qi+1, ..., c_qk} alors
16 |  |  |  |  |  |  si E_N ⊆ {c_qi, ..., c_qk} alors
17 |  |  |  |  |  |  |  E_i ← E_i \ {c_N};
18 |  |  |  |  |  |  fin si
19 |  |  |  |  |  fin si
20 |  |  |  |  fin pour tous
21 |  |  |  fin si
22 |  |  fin pour tous
23 |  fin tq
24 fin si
   Résultat : {C : (C, c, E) ∈ N}
```

Algorithme 2.1 : Recherche dans un arbre d'énumérations d'un ensemble.

Une fois les conflits minimaux ECM^+ établis, il suffit d'appliquer les résultats du théorème 2.1 pour établir les candidats minimaux. Ici encore DITO met à profit le solveur de CSP pour calculer les candidats minimaux à partir des conflits minimaux. La première condition du théorème 2.1 impose qu'un candidat minimal contient au moins un composant de tous les conflits minimaux (un candidat doit *couvrir* tous les conflits) et cet ensemble doit être minimal (seconde condition). On recherche ainsi l'en-

semble $EECM$ des ensembles minimaux couvrant les conflits minimaux. Ce problème de recherche est analogue au premier, on cherche des sous-parties d'un ensemble qui répondent à un critère (la couverture). La différence est dans le problème CSP qui traduit ici le problème de non-couverture des conflits minimaux par un ensemble de composants donné, il n'est satisfaisable pour une sous-partie donnée que si cette sous-partie ne couvre pas les conflits. On va donc explorer les nœuds d'un arbre comme précédemment jusqu'à trouver les sous parties minimales qui couvrent tous les conflits de ECM^+.

Les deux étapes sont illustrées par l'algorithme 2.2 qui est le point de départ de la mise en œuvre de l'outil Dito .

Entrées : problème PD
1 $ECM^+ \longleftarrow RechercheArbre(encode(\mathsf{PD}), \mathsf{COMPS})$;
2 $EECM \longleftarrow RechercheArbre(encode(ECM^+), \bigcup_{C \in ECM^+}(C))$;
Résultat : $\{\sigma(\Delta) : \Delta \in EECM\}$

Algorithme 2.2 : Algorithme de recherche des candidats minimaux.

Dans [38], on démontre que cet algorithme résout le problème du diagnostic à base de modèle en s'appuyant sur le théorème 2.1.

Théorème 2.2 *L'algorithme 2.2 est correct [38].*

Revenant sur l'exemple, Dito calcule les quatre candidats minimaux

$$\sigma(\mathsf{M3}), \sigma(\mathsf{A2}), \sigma(\mathsf{M1}, \mathsf{M2}), \sigma(\mathsf{A1}, \mathsf{M2})$$

du problème en 523ms et en utilisant 9Mo.

2.2.5 Stratégie de recherche des candidats minimaux

Intrinsèquement, la complexité de l'algorithme 2.2 est exponentielle en le nombre de composants de COMPS ce qui rend son utilisation directe en pratique impossible. Dans [38], on propose une stratégie de recherche des candidats minimaux s'appuyant sur la granularité du résultat. Supposons un instant que le système n'est composé que d'un seul composant $|\mathsf{COMPS}_1| = 1$, alors le problème de diagnostic $\mathsf{PD}_1 = (\mathsf{DS}_1, \mathsf{OBS}, \mathsf{COMPS}_1)$ est en fait un problème de détection (le système fonctionne-t-il correctement ou non ?). Si maintenant il est possible de *décomposer* ce système en deux sous-parties distinctes, à savoir deux composants d'intérêt $c_1, c_2 \in \mathsf{COMPS}_2$, alors on peut définir sur le même système un deuxième problème de diagnostic $\mathsf{PD}_2 = (\mathsf{DS}_2, \mathsf{OBS}, \mathsf{COMPS}_2)$. Le problème PD_2 quant à lui est un problème de localisation : il peut détecter si c_1 est anormal ou si c'est c_2 (voire les deux). Les problèmes PD_1 et

PD_2 s'appuient sur le même système et les mêmes observations, seule la granularité du problème diffère, PD_2 est un problème plus fin que PD_1 mais aussi un problème plus complexe sachant que $|COMPS_2| > |COMPS_1|$. Le principe de la stratégie adoptée est de chercher les candidats minimaux en privilégiant la résolution d'une successions de problèmes avec une complexité maîtrisée en contrôlant la cardinalité des différents ensembles de composants choisis et en allant de la résolution de problèmes grossiers vers la résolution de problèmes plus fins.

Le problème de diagnostic initial (définition 2.6) fixe les composants d'intérêt COMPS et définit ainsi la granularité maximale du résultat. Soit $\Pi \in 2^{COMPS}$ une partition de COMPS, on peut définir la description à la granularité de Π du système DS par DS_Π comme suit : on copie DS dans DS_Π où l'on substitue au prédicat An() le prédicat AnComp() et pour tout élément C (sous-ensemble de composants) de Π, on y ajoute la phrase :

$$\neg An(C) \Longrightarrow \bigwedge_{c \in C} \neg AnComp(c)$$

le problème de diagnostic de granularité Π devient le problème de diagnostic suivant :

$$PD_\Pi = (DS_\Pi, OBS, \Pi).$$

La recherche de candidats minimaux par une succession de problèmes de diagnostics granulaires est motivée par le résultat suivant. Soit $\Delta \subseteq COMPS$, on note $\mathcal{R}_\Pi(\Delta) \stackrel{\text{def}}{=} \{C \in \Pi : C \cap \Delta \neq \emptyset\}$ l'ensemble des éléments de Π couverts par Δ.

Théorème 2.3 ([38]) *Pour tout candidat minimal de $\sigma(\Delta_\Pi)$ issu de PD_Π il existe un ensemble non-vide de candidats minimaux $\sigma(\Delta)$ de PD tel que $\Delta_\Pi = \mathcal{R}_\Pi(\Delta)$.*

Ainsi, dès lors que l'on a déterminé un candidat minimal $\sigma(\Delta_\Pi)$ à la granularité de Π, on sait qu'il existe au moins un candidat minimal à la granularité maximale impliquant *uniquement* les composants inclus dans les ensembles appartenant à Δ_Π. On peut donc exonérer tout autre composant (on les suppose normaux) dans une nouvelle recherche de conflits plus granulaire tout en ayant la garantie que l'on aboutit à au moins un candidat minimal dans cette recherche. Le résultat suivant nous permet également d'affiner la recherche de successions de partitions.

Théorème 2.4 ([38] modifié) *Soit $\Pi = \{\pi_1, \ldots, \pi_k\}$, si π_i n'est impliqué dans aucun candidat minimal de PD_Π, alors :*

- *il n'y a pas de candidats minimaux $\sigma(\Delta)$ de PD impliquant* uniquement *des composants de π_i ;*
- *il peut y avoir des candidats minimaux $\sigma(\Delta)$ de PD impliquant des composants de π_i mais ces candidats doivent aussi impliquer des composants issus des candidats minimaux de PD_Π.*

La table 2.1 illustre une session de diagnostic avec cette stratégie sur un circuit électronique : le c6288. Ce circuit est un multiplieur 16 bits faisant partie de la liste des circuits ISCAS retenus par la communauté DX pour produire des résultats expérimentaux. Le c6288 a la particularité d'être un des plus grands circuits retenus. Le circuit a 2 bus d'entrée A et B (16 bits) et un bus de sortie P (32 bits). Ce circuit est composé de 2406 portes logiques (principalement des non-ous, des non-ets et des inverseurs) organisées en 16 colonnes d'additionneurs (C0, … , C15). Ce circuit est codé dans Dito avec l'aide de 11904 variables et de 11932 contraintes. L'outil CSP qui a été exploité par Dito est Choco [Team, 2008]. Pour cette expérimentation, une faute a été injectée dans la porte logique G7 de l'additionneur ad1 de la colonne C1, le nom complet de ce composant étant ainsi C1ad1G7. Sur ce système, on observe les entrées/sorties à savoir les bus A, B et P. Pour l'exemple de la table 2.1, les observations OBS sont :

$$\mathsf{OBS} \equiv (\mathsf{A} = 29449 \wedge \mathsf{B} = 17006 \wedge \mathsf{P} = 500809692).$$

Chaque ligne de la table représente un problème de diagnostic (du moins granulaire au plus granulaire). La ligne 0 représente le problème de détection. Les lignes suivantes s'appuient sur des partitions du plus en plus fines qui profitent des informations obtenues sur les partitions utilisées antérieurement. La ligne 10 indique que parmi les diagnostics candidats minimaux se trouve $\sigma(\mathrm{S}^2_{10})$ qui est l'état de santé réel du système après injection de la faute C1ad1G7. La présence de ce candidat illustre le fait que Dito est correct.

2.3 Diagnosticabilité des systèmes logiques

La section précédente a illustré différents problèmes de diagnostic de cohérence et a montré, dans ces cas précis, qu'il existait plusieurs candidats minimaux pour un problème donné et donc les observations disponibles appliquées à la connaissance du système DS ne suffisent pas à identifier avec certitude un seul candidat dans ces problèmes, il y a ambiguïté. Toutes les propriétés du système et de son observabilité offrant la garantie d'une solution unique pour un problème PD donné sont rassemblées autour de la notion de diagnosticabilité. La diagnosticabilité est une notion intrinsèque au système et à son observabilité. Dans le cas des systèmes logiques, il y a étonnamment eu peu de travaux sur le sujet [Console et al., 2000] contrairement au cas des autres types de systèmes qui seront présentés dans des chapitres ultérieurs. Cette section présente une définition de la notion de diagnosticabilité des systèmes logiques.

2.3.1 Contexte des travaux

Le travail présenté ici a été réalisé dans le cadre du projet européen WS-Diamond ([1], [2], [3], [18]). Des partenaires de ce projet ont proposé de développer une méthode

	Partition	Solution	Temps(ms)
0	$\{S_0^0 = \{\mathsf{COMPS}\}\}$	$\sigma(S_0^0)$	114670
1	$\{S_1^0 = \{\mathsf{C0}\}, S_1^1 = \{\mathsf{C1}, \dots, \mathsf{C8}\}, S_1^2 = \{\mathsf{C9}, \dots, \mathsf{C15}\}\}$	$\sigma(S_1^1)$	285408
2	$\{S_2^0 = \{\mathsf{C1}, \dots, \mathsf{C3}\}, S_2^1 = \{\mathsf{C4}, \mathsf{C5}\}, S_2^2 = \{\mathsf{C6}, \dots, \mathsf{C8}\}$	$\sigma(S_2^0)$	131735
3	$\{S_3^0 = \{\mathsf{C1}\}, S_3^1 = \{\mathsf{C2}\}, S_3^2 = \{\mathsf{C3}\}\}$	$\sigma(S_3^0)$	56098
4	$\{S_4^0 = \{\mathsf{C1ad1}, \dots, \mathsf{C1ad15}\}, S_4^1 = \{\mathsf{C1gi1}, \dots, \mathsf{C1gi15}, \mathsf{C1gj0}\}\}$	$\sigma(S_4^0), \sigma(S_4^1)$	22850
5	$\{S_5^0 = \{\mathsf{C1ad1..8}, \mathsf{C1gi1..8}\}, S_5^1 = \{\mathsf{C1ad9..16}, \mathsf{C1gi9..16}\}$	$\sigma(S_5^0)$	57195
6	$\{S_6^0 = \{\mathsf{C1ad1..2}, \mathsf{C1gi1..2}\}, \dots, S_6^3 = \{\mathsf{C1ad7..8}, \mathsf{C1gi7..8}\}\}$	$\sigma(S_6^0)$	33760
7	$\{S_7^0 = \{\mathsf{C1ad1}\}, S_7^1 = \{\mathsf{C1ad2}\}, S_7^2 = \{\mathsf{C1gi1}\}, S_7^3 = \{\mathsf{C1gi2}\}\}$	$\sigma(S_7^0)$, $\sigma(S_7^2)$	36861
8	$\{S_8^0 = \{\mathsf{C1ad1G1..4}\}, S_8^1 = \{\mathsf{C1ad1G5..9}\}\}$	$\sigma(S_8^0)$, $\sigma(S_8^1)$	17373
9	$\{S_9^0 = \{\mathsf{C1ad1G1}\}, \dots S_9^3 = \{\mathsf{C1ad1G4}\}\}$	$\sigma(S_9^1), \sigma(S_9^2), \sigma(S_9^3)$	37498
10	$\{S_{10}^0 = \{\mathsf{C1ad1G5}\}, \dots S_{10}^4 = \{\mathsf{C1ad1G9}\}\}$	$\sigma(S_{10}^1), \sigma(\mathbf{S_{10}^2}), \sigma(S_{10}^3)$	66456

TABLE 2.1 – Session de diagnostic sur le circuit c6288 (circuit ISCAS).

de diagnostic pour les services Web s'appuyant sur la définition 2.6 du diagnostic sur les systèmes logiques. L'objectif, au sein de notre équipe de recherche, était de définir les propriétés que les systèmes devaient posséder afin qu'une décision de réparation du service défaillant soit effective à partir des conclusions données par l'outil de diagnostic de notre partenaire. Cette étude de diagnosticabilité a été conduite dans le cadre des travaux de thèse de Xavier Pucel sous la direction de Louise Travé-Massuyès. J'ai participé à la mise au point de cette définition en m'appuyant sur mon expérience de ces problèmes dans le cadre des systèmes à événements discrets.

2.3.2 Notion de diagnosticabilité dans les systèmes logiques

On se place ici dans le cadre général de la définition 2.6 avec l'introduction de modes de fonctionnement sur les composants du type $\mathsf{An_M}(,)$ (voir page 20). Pour simplifier les notations sans perte de généralité, on introduit ici dans DS une nouvelle variable propositionnelle $\mathsf{f_i}$ associée à un mode $\mathsf{M_{j_i}}$ d'un composant $\mathsf{c_{j_i}} \in \mathsf{COMPS}$ telle que :

$$\mathsf{f_i} \equiv \mathsf{An_M}(\mathsf{c_{j_i}}, \mathsf{M_{j_i}}).$$

Soit F l'ensemble des variables de mode $\mathsf{F_i}$, l'état de santé du système peut alors être décrit par une conjonction de littéraux de F :

$$\sigma(\Delta) = \bigwedge_{\mathsf{F_i} \in \Delta} \mathsf{F_i} \wedge \bigwedge_{\mathsf{F_i} \in \mathcal{F} \setminus \Delta} \neg \mathsf{F_i}, \Delta \subseteq \mathsf{F}.$$

L'ensemble Δ décrit ainsi un ensemble de modes de dysfonctionnement. Soit maintenant $\mathsf{O_{DS}}$ l'ensemble des faits potentiellement observables sur le système DS, autrement dit, $(\mathsf{DS}, \mathsf{F}, \mathsf{OBS})$ est un problème de diagnostic si et seulement si $\mathsf{OBS} \in \mathsf{O_{DS}}$. On fera l'hypothèse ici qu'aucune observation de $\mathsf{O_{DS}}$ n'en subsume une autre ($\forall \mathsf{OBS}_1, \mathsf{OBS}_2 \in \mathsf{O_{DS}}, \mathsf{OBS}_1 \nVdash \mathsf{OBS}_2$).

Définition 2.14 (Signature logique de mode) *La signature d'un mode* f *est l'ensemble des faits observables* $\mathsf{OBS} \in \mathsf{O_{DS}}$ *tels qu'il existe au moins un ensemble* $\Delta \subseteq \mathsf{F}$: $\mathsf{f} \in \Delta$ *pour lequel* $\sigma(\Delta)$ *est un diagnostic candidat du problème* $(\mathsf{DS}, \mathsf{F}, \mathsf{OBS})$. *La signature du mode* f *est notée* $Sig(\mathsf{f})$.

Intuitivement, si OBS est dans la signature $Sig(\mathsf{f})$ alors il est toujours *possible* d'expliquer ce comportement en suspectant que le système subit, parmi d'autres, le mode de dysfonctionnement f. Cette notion de signature est traduite à partir de celle issue des travaux de [Travé-Massuyès et al., 2006] et [Cordier et al., 2006].

Soit $\mathsf{ok} \stackrel{\text{def}}{=} \bigwedge_{\mathsf{f} \in \mathsf{F}} \neg \mathsf{f}$ la variable propositionnelle attestant l'état de santé normal du système, on peut également lui associer une signature comme un autre mode :

$$Sig(\mathsf{ok}) = \{\mathsf{OBS} \in \mathsf{O_{DS}} \ : \ \sigma(\emptyset) \text{ est un diagnostic candidat.}\}$$

À partir de cette notion de signature, on définit une première notion de diagnosticabilité sur les systèmes logiques.

Définition 2.15 (Diagnosticabilité statique de mode) *Un système logique est statiquement diagnosticable si :*

$$\forall \mathsf{f}_1 \in \mathsf{F} \cup \{\mathsf{ok}\}, \forall \mathsf{f}_2 \in \mathsf{F} \cup \{\mathsf{ok}\}, \mathsf{f}_1 \neq \mathsf{f}_2 \Longrightarrow Sig(\mathsf{f}_1) \cap Sig(\mathsf{f}_2) = \emptyset.$$

Dès lors que la condition de diagnosticabilité est remplie dans un système, tout fait observable OBS appartenant à la signature d'un mode f n'appartient à aucune autre signature. Dans ces conditions, l'hypothèse que c'est le mode f qui produit OBS n'est plus une possibilité, c'est devenu une certitude. Cette première définition impose une condition qui garantit ainsi que pour toute observation OBS, le mode de dysfonctionnement du système sera parfaitement identifié à l'aide d'un algorithme de diagnostic résolvant le problème $\mathsf{PD} = (\mathsf{DS}, \mathsf{F}, \mathsf{OBS})$. Néanmoins, cette condition est restrictive, elle impose notamment que le système ne peut subir qu'un seul mode de dysfonctionnement en même temps. Cette condition n'est pas applicable dès lors que l'on considère la possibilité que le système puisse subir un ensemble de modes de dysfonctionnement en même temps. Ainsi cette diagnosticabilité n'est pas assez générale notamment quand on cherche à garantir que l'on peut toujours déterminer que le ou les modes de dysfonctionnement courant du système font toujours partie d'un sur-ensemble de modes particulier. Cette propriété est notamment intéressante dans l'analyse des systèmes

autoguérissants (voir la section 5.1 page 153 sur le sujet) où l'on y utilise la notion de macromode.

Définition 2.16 (Macromode) *Un macromode est une disjonction de modes :*

$$\mathsf{mf} = \mathsf{f}_{i_1} \vee \cdots \vee \mathsf{f}_{i_n}, \mathsf{f}_{i_j} \in \mathsf{F}.$$

Un macromode offre un moyen de régler les conditions de diagnosticabilité plus finement. En introduisant un macromode $\mathsf{mf} = \mathsf{f}_{i_1} \vee \cdots \vee \mathsf{f}_{i_n}$ dans le modèle, on affirme pour des raisons propres à l'application considérée, qu'il n'est pas nécessaire d'être en mesure de pouvoir discriminer entre les modes f_{i_1} et que la simple garantie, qu'à partir des observations on peut toujours affirmer/infirmer la présence du macromode mf, est suffisante.

Définition 2.17 (Signature logique de macromode) *La signature d'un macromode* $\mathsf{mf} = \mathsf{f}_{i_1} \vee \cdots \vee \mathsf{f}_{i_n}, \mathsf{f}_{i_j} \in \mathsf{F}$ *est l'ensemble des faits observables* $\mathsf{OBS} \in \mathsf{O}_{\mathsf{DS}}$ *tels qu'il existe au moins un ensemble* $\Delta \subseteq \mathsf{F} : \Delta \cap \{\mathsf{f}_{i_1}, \ldots, \mathsf{f}_{i_n}\} \neq \emptyset$ *pour lequel* $\sigma(\Delta)$ *est un diagnostic candidat du problème* $(\mathsf{DS}, \mathsf{F}, \mathsf{OBS})$.

On serait tenté avec la notion de signature des macromodes de définir la diagnosticabilité des macromodes comme celle de la diagnosticabilité de mode (définition 2.15) en imposant une intersection vide sur les signatures. Néanmoins, il ne faut pas oublier de mentionner que deux macromodes mf_1 et mf_2 peuvent impliquer un ensemble commun de modes, par exemple :

$$\mathsf{mf}_1 = \mathsf{f}_1 \vee \mathsf{f}_2 \vee \mathsf{f}_3, \mathsf{mf}_2 = \mathsf{f}_2 \vee \mathsf{f}_3 \vee \mathsf{f}_4.$$

Imposer un telle définition de diagnosticabilité aurait pour conséquence dans l'exemple précédent d'*interdire* la présence des modes f_2 et f_3 dans un système diagnosticable ce qui est un contresens. La notion de diagnosticabilité ainsi introduite s'appuie sur un sous-ensemble de cette signature.

Définition 2.18 (Signature caractéristique d'un macromode) *Une signature caractéristique d'un macromode* mf *est un ensemble* $Sig_c(\mathsf{mf})$ *tel que :*

$$Sig_c(\mathsf{mf}) \subseteq Sig(\mathsf{mf}) \setminus (Sig(\mathsf{ok}) \cup \bigcup_{\mathsf{f} \notin \mathsf{mf}} Sig(\mathsf{f})).$$

À partir de toute observation OBS issue d'une signature $Sig_c(\mathsf{mf})$ de mf, le diagnostic peut garantir :

1. le macromode mf a nécessairement eu lieu car seuls les modes impliqués dans mf ont pu avoir lieu.

2. un autre macromode mf' impliquant un ensemble de modes communs à mf a pu avoir lieu.

La condition de diagnosticabilité doit maintenant imposer qu'il ne doit pas y avoir d'ambiguïté entre les macromodes mf et mf', ce qui impose que les signatures caractéristiques de chaque macromode soient disjointes pour garantir une décision sur le macromode suspecté.

Définition 2.19 (Diagnosticabilité statique de macromode) *Soit* MF *un ensemble de macromodes tel que pour tout mode* $\mathsf{f} \in \mathsf{F}$*, il existe au moins* $\mathsf{mf} \in \mathsf{MF}, \mathsf{f} \models \mathsf{mf}$*, l'ensemble* MF *est diagnosticable sur le système* DS *s'il existe une partition de signatures caractéristiques* $\{Sig_c(\mathsf{mf})\}_{\mathsf{mf}\in\mathsf{MF}}$ *de* O_{DS}.

Cette deuxième définition étend la première, ce qui est décrit par le résultat suivant.

Proposition 2.1 ([51]) *Soit* $\mathsf{F} = \{\mathsf{f}_1, \dots, \mathsf{f}_n\}$ *l'ensemble des modes de dysfonctionnement d'un système modélisé par* DS*, le système est diagnosticable au sens de la définition 2.15 si et seulement si l'ensemble* $\mathsf{MF} = \{\mathsf{mf}_1, \dots, \mathsf{mf}_n\}$ *où* $\mathsf{mf}_i \equiv \mathsf{f}_i$, $i \in \{1, \dots, n\}$ *est diagnosticable au sens de la définition 2.19.*

Mais dorénavant, on peut également étudier sur le système la diagnosticabilité sur des ensembles de macromodes différents et plus complexes afin d'inférer des garanties précises sur la performance d'une activité de diagnostic sur ce système.

2.3.3 Diagnosticabilité revisitée

La mise en place des définitions précédentes repose en premier lieu sur le problème de diagnostic (peut-on diagnostiquer avec certitude un mode, ou un macromode ?). On fixe donc a priori la propriété que l'on cherche à diagnostiquer. Une autre façon de procéder pour la définition de la diagnosticabilité est de procéder par nécessité. Qu'est-il nécessaire d'avoir dans le système pour qu'une propriété quelconque (mode, macromode ou toute autre chose) soit diagnosticable ? Une propriété de DS est une phrase logique φ telle que $\mathsf{DS} \wedge \varphi$ est satisfaisable.

Définition 2.20 (Bloc diagnosticable [50]) *Soit* OBS *une observation possible de* O_{DS}*, un bloc diagnosticable de* DS *est l'ensemble des propriétés* φ *tel que :*

$$\varphi \models \mathsf{OBS}.$$

Le bloc diagnosticable de OBS *est la formule :*

$$\mathsf{BD}(\mathsf{OBS}) = \bigvee_{\varphi : \varphi \models \mathsf{OBS}} \varphi.$$

La notion de bloc diagnosticable est intrinsèque au système et est indépendante de la propriété que l'on cherche effectivement à diagnostiquer. Par définition, dès lors que le système DS émet l'observation OBS, toute propriété φ_1 ne pourra être distinguée de toute autre propriété φ_2 du bloc diagnosticable de OBS. Du point de vue de la diagnosticabilité, un bloc diagnosticable est atomique, toute propriété diagnosticable est donc nécessairement une disjonction de blocs diagnosticables.

Définition 2.21 (Diagnosticabilité de propriété [50]) *Soit φ une propriété de* DS*, φ est diagnosticable si*

$$\varphi \equiv \bigvee_{\mathsf{OBS} \in \mathsf{O_{DS}} : \varphi \wedge \mathsf{OBS} \text{ satisfaisable}} \mathsf{BD(OBS)}.$$

Cette nouvelle définition, indépendante de l'objectif de diagnostic, généralise automatiquement la première.

Proposition 2.2 ([50]) *Soit* $\mathsf{F} = \{\mathsf{f_1}, \ldots, \mathsf{f_n}\}$ *l'ensemble des modes de dysfonctionnement d'un système modélisé par* DS*, le système est diagnosticable au sens de la définition 2.15 si et seulement si l'ensemble des propriétés* $\varphi_i \equiv \mathsf{f_i}, i \in \{1, \ldots, n\}$ *sont diagnosticables au sens de la définition 2.21.*

Ce qui est vrai pour la définition 2.15 ne l'est pas pour la définition 2.19 qui s'appuie sur les macromodes. En effet, deux macromodes $\mathsf{mf_1}$ et $\mathsf{mf_2}$ peuvent être diagnosticables au sens de la définition 2.19 sans pour autant l'être au sens de la définition 2.21. La raison de cette différence est parfaitement visible dès lors que l'on considère deux macromodes $\mathsf{mf_1}$ et $\mathsf{mf_2}$ qui impliquent un ensemble de modes communs non vide. Supposons par exemple que cet ensemble implique le mode f. Une partie de la signature du mode f peut faire partie de la signature caractéristique de $\mathsf{mf_1}$ et une autre peut faire partie de la signature caractéristique de $\mathsf{mf_2}$. Le principe de la définition 2.18 repose alors sur le fait que si OBS appartient à la signature de f, il peut y avoir ambiguïté sur le macromode mais en aucun cas sur le mode (le mode f a lieu, soit représenté par le macromode $\mathsf{mf_1}$, soit par le macromode $\mathsf{mf_2}$). Une autre façon de voir les choses est de dire que, dès que le mode f a lieu et que le système produit une observation qui appartient à la signature caractéristique de $\mathsf{mf_2}$ tout en sachant qu'elle aurait très bien pu faire partie de la signature caractéristique de $\mathsf{mf_1}$ (car c'est un choix), on dit que $\mathsf{mf_2}$ *préempte* $\mathsf{mf_1}$: la décision afférente à la présence de $\mathsf{f_1}$ sera fonction de la décision afférente à la présence de $\mathsf{mf_2}$ sachant que $\mathsf{f_1}$ est un mode possible dans $\mathsf{mf_2}$.

Définition 2.22 (Préemptabilité d'une propriété [50]) *Soit φ une propriété de* DS *et Φ un ensemble de propriétés tel que $\Phi \cap \{\varphi\} = \emptyset$, φ est préemptable par Φ si pour toute observation* $\mathsf{OBS} \in \mathsf{O_{DS}}$ *telle que $\varphi \wedge$* OBS *est satisfaisable et* $\mathsf{BD(OBS)} \nvdash \varphi$ *on a :*

$$\exists \varphi' \in \Phi, \mathsf{BD(OBS)} \vDash \varphi'.$$

La définition 2.22 généralise ainsi le principe énoncé sur les macromodes à des propriétés quelconques. Pour une observation OBS donnée, le diagnostic conduit à émettre des hypothèses de diagnostic en s'appuyant sur le bloc diagnosticable BD(OBS) qui rend φ' toujours vrai et qui rend φ parfois vrai. Sur l'observation OBS, φ est ainsi préemptable par φ'.

Définition 2.23 (Diagnosticabilité d'un ensemble de propriétés [50]) *Soit* Φ *un ensemble de propriétés,* Φ *est diagnosticable sur* DS *si toute propriété* $\varphi \in \Phi$ *est soit diagnosticable, soit préemptable par* $\Phi \setminus \{\varphi\}$.

Cette définition est la plus générale connue à ce jour. Elle englobe notamment la définition sur les macromodes.

Proposition 2.3 *Soit* MF *un ensemble de macromodes, l'ensemble* MF *est diagnosticable sur le système* DS *au sens de la définition 2.19 si et seulement si il l'est au sens de la définition 2.23.*

2.4 Théorie logique du méta-diagnostic

Comme nous l'avons vu dans les sections précédentes, le diagnostic à base de modèles repose sur la différentiation entre le modèle (DS, COMPS) qui est une construction intellectuelle de la réalité d'un système et un moteur de diagnostic générique et indépendant de tout modèle qui va résoudre un problème PD = (DS, COMPS, OBS) pour un modèle (DS, COMPS) donné. Le diagnostic à base de modèle repose également sur des hypothèses fortes, que l'on peut résumer ici de façon intuitive comme suit :

— le modèle représente *effectivement* la réalité [Struss, 1991] ;

— le moteur de diagnostic résout *effectivement* le problème pour un modèle donné.

La mise en place dans un cadre réaliste d'un tel paradigme est complexe. Les difficultés liées au coût de modélisation, de mise en œuvre, aux freins technologiques et à la capacité d'instrumentation d'un système font que en pratique de nombreux outils de diagnostic sont mis en œuvre par des *approximations* invalidant les hypothèses fondamentales du diagnostic à base de modèles. Les questions relatives à la performance d'un outil de diagnostic face à un modèle sont répertoriées dans cette section. Cette section présente ainsi une théorie logique pour le raisonnement sur les performances d'un système de diagnostic. Ce raisonnement étant lui-même un raisonnement diagnostique, cette théorie est communément appelée la théorie du méta-diagnostic.

2.4.1 Contexte des travaux

Les travaux sur la théorie du méta-diagnostic ont été initiés et développés dans le cadre des travaux de thèse de Nuno Belard, étudiant encadré par Michel Combacau

et moi-même. Ce travail s'insère dans le cadre d'une convention CIFRE avec Airbus. Nuno Belard faisait partie du personnel Airbus et avait intégré le service de maintenance d'Airbus. Ce fait a été essentiel dans le cadre de ses travaux car Nuno Belard a pu avoir accès à des données industrielles confidentielles et a pu ainsi motiver son travail académique sur des problématiques industrielles réelles.

2.4.2 Système de diagnostic

La motivation originale de ce travail est un constat industriel [Cheriere, 2009], [Hepes et al., 2009]. La société Airbus construit plusieurs types d'avions (A320, A350, A380, …). Chaque type d'avion dispose à son bord d'un système de surveillance et de diagnostic qui lui est propre : ce système est composé d'un ensemble de calculateurs sur lesquels sont déployés des outils logiciels. Les outils logiciels (à travers ceux là les algorithmes qu'ils mettent en œuvre) sont en général différents d'un type d'avion à un autre (mais pas toujours) ou bien même dans deux avions du même type. L'une des principales raisons à cela est que chaque type d'avion est généralement développé dans un programme qui lui est propre avec des choix technologiques qui lui sont propres. La maintenance de ces appareils ayant un coût important, elle se doit d'être efficace or un frein à cette efficacité est la diversité des appareils et de leur capacité d'auto-diagnostic qui est un obstacle à la mutualisation des moyens pour la maintenance. En terme de maintenance, on peut ainsi se poser la question suivante : quelles sont les performances d'un système de diagnostic par rapport à un autre ? En quoi un système de diagnostic est-il meilleur qu'un autre ? Et par ailleurs, que signifie *meilleur* ? La théorie du méta-diagnostic répond formellement à ces questions.

L'objet de l'étude du méta-diagnostic est le *système de diagnostic*, à savoir l'ensemble des éléments mis en œuvre (capteurs, calculateurs, réseaux de communication, logiciels) pour établir un diagnostic sur un système complexe (dans ce cas d'étude : un avion). Ce déploiement d'équipements et de logiciels réalisent un *algorithme de diagnostic*.

Définition 2.24 (Algorithme de diagnostic) *Un algorithme de diagnostic* ALG *est un algorithme dont l'objectif est de résoudre un problème de diagnostic* PD.

Cette première définition est informelle mais elle a toute son importance. Il faut voir ici un algorithme de diagnostic ALG comme une *intention*, une *tentative* de résoudre le problème du diagnostic mais on ne dit pas ici que ALG répond effectivement au problème. L'algorithme ALG peut donc être défaillant.

Dans le cadre de cette étude, on s'est limité à la vision statique du diagnostic (résolution de problème du type $\mathsf{PD} = (\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS}_\mathsf{C}, \mathsf{OBS}_\mathsf{S})$, voir définition 2.6). Dans ce cadre, on peut voir tout algorithme ALG de la manière suivante.

Proposition 2.4 (Simulation d'algorithme) *Tout algorithme de diagnostic* ALG *peut être associé à une simulation* (EES, PTC, PTE) *où :*

— EES *est un énumérateur d'états de santé du système* $\mathsf{EES} \subseteq 2^{\mathsf{COMPS}}$ *;*

— PTC *est un prouveur de théorème pour la cohérence avec* $\mathsf{OBS_C}$ *;*

— PTE *est un prouveur de théorème pour l'explication de* $\mathsf{OBS_S}$.

Avec cette proposition, on dit la chose suivante. Si on considère le problème de diagnostic suivant $\mathsf{PD} = (\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS_C}, \mathsf{OBS_S})$ et un algorithme de diagnostic ALG, cet algorithme va fournir un ensemble de solutions (qu'elles soient correctes ou non). Ces solutions sont des ALG-candidats.

Définition 2.25 (ALG**-candidat)** *L'état de santé* $\sigma(\Delta)$ *est un* ALG*-candidat de* PD *(noté* $\mathsf{PD} \vdash_{\mathsf{ALG}} \sigma(\Delta)$*) si :*

1. $\Delta \in \mathsf{EES}$ *;*
2. $\mathsf{DS} \wedge \mathsf{OBS_C} \nvdash_{\mathsf{PTC}} \neg\sigma(\Delta)$ *;*
3. $\mathsf{DS} \wedge \mathsf{OBS_C} \wedge \sigma(\Delta) \vdash_{\mathsf{PTE}} \mathsf{OBS_S}$.

Supposons en effet que $\sigma(\Delta)$ soit une solution proposée par ALG alors Δ est nécessairement dans EES sinon ALG n'aurait même pas vu, et donc pas traité cette hypothèse de défaillance. $\sigma(\Delta)$ étant une solution selon ALG il répond donc aux conditions requises par la définition 2.7 mais qui est mise en œuvre par la logique interne de ALG simulée par les prouveurs de théorèmes PTC et PTE.

C'est en ce sens que le triplet (EES, PTC, PTE) simule l'algorithme ALG. En particulier PTC simule la logique de cohérence entre le modèle et les observations de l'algorithme ALG et PTE simule le raisonnement abductif interne de ALG pour l'explication des symptômes $\mathsf{OBS_S}$. Δ est ainsi solution du problème $\mathsf{PD} = (\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS_C}, \mathsf{OBS_S})$ selon ALG si PTC *croît* en la cohérence des observations avec $\sigma(\Delta)$ et si PTE *croît* expliquer les observations symptomatiques $\mathsf{OBS_S}$.

Pour illustrer cette simulation sur un algorithme de diagnostic concret, il suffit de reprendre l'algorithme utilisé dans Dito . L'énumérateur d'états de Dito est implicitement mis en œuvre par la recherche de conflits minimaux et le théorème 2.1 qui permet de déterminer la liste des candidats minimaux. Chaque nœud de l'arbre énumérant les conflits potentiels (voir figure 2.4) représente implicitement un ensemble d'états de santé candidats potentiels (qui sont donc dans EES). Dito garantit la détermination de tous les candidats minimaux (et donc implicitement tous les candidats) par une énumération implicite de toutes les possibilités :

$$\mathsf{EES}_{\text{Dito}} = 2^{\mathsf{COMPS}}.$$

La logique de cohérence de Dito est mise en œuvre par la programmation de CSP, reprenant la ligne 11

$$Sol(CSP') = \emptyset$$

de l'algorithme 2.1 lorsqu'il est appelé pour la recherche des ensembles couvrants (ligne 2 de l'algorithme 2.2), alors cette ligne teste exactement :

$$\mathsf{DS} \wedge \mathsf{OBS}_\mathsf{C} \nvdash_{\mathsf{PTC}} \neg\sigma(\Delta)$$

par un test de satisfaisabilité sur $\mathsf{DS} \wedge \mathsf{OBS}_\mathsf{C} \wedge \sigma(\Delta)$. Concernant la partie abductive, comme DITO ne met en œuvre qu'un diagnostic de cohérence $\mathsf{OBS}_\mathsf{S} \equiv \bot$.

Un autre exemple est l'algorithme initial *Diagnose* [Reiter, 1987]. Il est explicitement décrit comme la recherche de candidats dans un espace de candidats à l'aide d'un prouveur de théorème. En ce sens, l'algorithme *Diagnose* est son propre simulateur. On peut également citer le moteur de diagnostic générique GDE [deKleer and Williams, 1987] basé sur des systèmes de maintenance d'hypothèses (ATM : assumption-based truth maintenance [Forbus and deKleer, 1993]).

L'objet d'étude du méta-diagnostic est le système de diagnostic défini ainsi.

Définition 2.26 (Système de diagnostic [8]) *Un* système de diagnostic *est un quintuplet* $(\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS}_\mathsf{C}, \mathsf{OBS}_\mathsf{S}, \mathsf{ALG})$ *où :*

— $(\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS}_\mathsf{C}, \mathsf{OBS}_\mathsf{S})$ *est un problème de diagnostic ;*

— ALG *est un algorithme de diagnostic.*

2.4.3 Sur les performances d'un système de diagnostic

La question qui est maintenant posée est de déterminer les raisons pour lesquelles un système de diagnostic peut être défaillant. Pour cela, il faut d'abord chercher les raisons pour lesquelles on a l'assurance que le système de diagnostic retourne le résultat attendu.

2.4.3.1 Objectifs d'un système de diagnostic

Étant donné un système réel, on peut lui associer son état de santé réel, on le note σ_T. L'objectif ultime de tout algorithme de diagnostic est de déterminer σ_T et uniquement σ_T.

Définition 2.27 (validité [5]) *Un système de diagnostic sur un problème* PD *est valide si son diagnostic contient* σ_T.

La notion de validité est une notion évidente mais elle est ici d'une très grande importance et une source majeure de problème dans les systèmes de diagnostic en général. C'est l'une d'ailleurs des difficultés majeures du raisonnement diagnostic contrairement à des raisonnements de type prédictif (planification, commande). En planification/commande, la validation est plus aisée : tant que les réalisations des plans/lois de commandes générées par le planificateur sur le système sont compatibles avec les

objectifs initiaux (les buts, les consignes), on considère le plan (et donc le planificateur) comme valide. En diagnostic, comment en effet obtenir la validité d'un résultat issu d'un système de diagnostic alors qu'il n'existe aucun autre moyen simple (comme l'observation de l'exécution d'un plan) d'obtenir ce résultat σ_{T}.

Définition 2.28 (certitude[7]) *Un système de diagnostic sur un problème* PD *est certain si*

$$|\{\sigma(\Delta) : \mathsf{ALG}\text{-}candidat\}| = 1.$$

La notion de certitude est naturellement liée aux conditions de diagnosticabilité (voir section 2.3) mais pas uniquement. Un algorithme ALG peut très bien ne pas profiter de toute l'observabilité du système et générer ainsi encore plus d'ambiguïté.

2.4.3.2 Prérequis sur la connaissance

Le premier prérequis pour un système de diagnostic non défaillant est la satisfaisabilité du modèle.

Définition 2.29 (Satisfaisabilité du modèle) *Le modèle* (DS, COMPS) *est satisfaisable s'il existe au moins un modèle dans la théorie* DS.

Le prérequis de satisfaisabilité de DS peut sembler très naïf mais il ne l'est pas. Dans des systèmes complexes et hétérogènes tels qu'un système aéronautique, DS est constitué d'un agrégat de nombreuses phrases logiques (règles) qui ont été définies/extraites par des moyens différents dans des formats différents. Ainsi la source du problème de diagnostic peut très bien venir du fait que le modèle utilisé n'est déjà pas logiquement cohérent et invalide ainsi toute démarche de diagnostic.

L'objectif de toute science est de définir des descriptions formelles (modèles mathématiques) qui expliquent, prédisent la réalité. Si l'on s'appuie sur l'hypothèse de l'existence de tels modèles [1], on peut alors affirmer que ces descriptions sont *ontologiquement vraies*. En se fondant sur la théorie des modèles [Hodges, 1993], [Marker, 2002], [Hedman, 2004] qui formalise la notion de structure d'information, et les notions d'interprétation et d'extensions sur ces structures, cette propriété peut s'énoncer de la façon suivante.

Définition 2.30 (Vérité ontologique [8]) *Soit* Ω *l'ensemble des structures,* $\Psi \in \Omega$ *la structure d'information, une théorie* T *est* ontologiquement vraie *s'il existe un modèle de* T *qui soit une interprétation correcte dans une structure* $\mathcal{P}$ *dont une extension* $\mathcal{R}$ *est isomorphe à* Ψ.

1. Nous ne nous limiterons ici qu'à cette hypothèse et n'approfondirons pas les considérations philosophiques de cette question bien que les travaux décrits ici aient pu provoquer de nombreux débats sur cette question entre nous.

Cette propriété qui est informellement intuitive (tout ce qui est logiquement décrit est vrai [Tarski, 1983]) semble évidente mais elle a une importance particulière en diagnostic pour deux raisons. Premièrement, le diagnostic est une activité où l'on confronte l'observation réelle du système (et non une simulation) à une description formelle du système. La deuxième raison est la nature intrinsèque du raisonnement diagnostic qui est abductive. Les explications issues du diagnostic proviennent du modèle ce qui rend leur validation particulièrement délicate car il est souvent impossible par la simple mesure du système de valider une explication. De par ces deux raisons, on distingue deux prérequis sur la connaissance : la vérité ontologique des observations (OBS est une théorie) et la vérité ontologique de la description du système qui est définie comme suit.

Définition 2.31 (Modèle du système ontologiquement vrai [8]) *Soit* (DS, COMPS) *un modèle de système, il est ontologiquement vrai si pour toute observation* OBS *ontologiquement vraie, la théorie* DS ∧ OBS *est ontologiquement vraie.*

La définition précédente prend en compte la difficulté de modélisation de DS [Weld, 1992]. En effet, on pourrait restreindre et requérir que DS soit une théorie ontologiquement vraie. Néanmoins, le raisonnement diagnostic étant toujours la confrontation d'une observation OBS à cette théorie, on ne demande que la vérité ontologique de la théorie DS ∧ OBS, DS peut contenir des interprétations non vraies mais qui sont incohérentes avec toute observation ontologiquement vraie.

Définition 2.32 (Complétude du modèle[8]) *Le modèle* DS *est complet si pour tout contexte* CXT *de partition* (P_E, P_s) *de* DS*, pour toute conjonction du format*

$$\varphi \equiv \bigwedge_{p_i \in P_E} v(p_i) = v_i,$$

et pour tout état de santé possible $\sigma(\Delta)$ *si* DS ∧ CXT ∧ φ ∧ $\sigma(\Delta)$ *est satisfaisable alors* DS ∧ CXT ∧ φ ∧ $\sigma(\Delta)$ *est une théorie complète.*

Intuitivement, on dit que DS est complet s'il est déterministe. Pour un contexte donné, la fonction du système est totalement décrite, en particulier les paramètres de sortie sont déterminés sans ambiguïté à partir des paramètres d'entrée du système. Cette notion de complétude est liée ici à la notion de théorie complète, elle n'indique donc en rien si la théorie en question est ontologiquement vraie, en particulier elle n'indique aucunement que DS modélise tous les comportements du système, c'est une notion différente : la notion de couverture.

Définition 2.33 (Couverture[5]) *Soit* (DS, COMPS) *un modèle de système, il couvre le système :*

— structurellement *si l'ensemble des paramètres* p *contient l'ensemble des paramètres réels du système ;*
— contextuellement *si l'ensemble des contextes dans* (DS, COMPS) *inclut l'ensemble des contextes réels.*

Derrière la notion de couverture, on y trouve les problèmes d'absence de connaissance sur un système. Certains contextes et comportements associés ne sont pas connus bien qu'ils existent [Console et al., 1989], [Yeung and Kwong, 2005]. Si un système n'est pas couvert structurellement par le modèle, cela signifie en général que la fonction même du système n'est pas connue entièrement ou que la connaissance que l'on en a est structurellement erronée. Si le système est couvert structurellement mais non-couvert contextuellement, la fonction du système n'est donc pas entièrement connue, certains modes opératoires ne sont pas identifiés.

2.4.3.3 Prérequis sur le problème de diagnostic

Le sous-section précédente propose une liste de propriétés sur la connaissance du système qui sont indépendantes du problème de diagnostic mais qui sont propres à un problème de modélisation. Mais il existe des sources de défaillances qui sont spécifiques aux problèmes de diagnostic.

Définition 2.34 (Satisfaisabilité du problème) *Le problème* PD = (DS, COMPS, OBS) *est satisfaisable si* DS ∧ COMPS ∧ OBS *est satisfaisable.*

Même si le modèle (DS, COMPS) est satisfaisable (voir définition 2.29), il est tout à fait possible que sa confrontation avec OBS soit non-satisfaisable. Cette propriété indique qu'il existe une incohérence entre l'observation du système et la connaissance a priori que l'on a du système. Les conséquences d'un problème non-satisfaisable sont immédiates, tout algorithme « correct » retournera bien évidement un diagnostic nul (pas de candidats possibles).

Définition 2.35 (Problème contextuellement observable) *Le problème* PD = (DS, COMPS, OBS) *est contextuellement observable si* DS ∧ OBS ⊨ CXT. *Cette observabilité est faible si seuls les paramètres d'entrée de* CXT *sont observés, elle est forte si tous les paramètres d'entrée et de sortie de* CXT *sont observés.*

Cette deuxième notion est liée à la qualité des observations. En toute généralité, le système a plusieurs contextes (il n'en a pas qu'un seul qui serait donc totalement déterminé dans DS, dans ce dernier cas le problème serait bien évidemment contextuellement observable). Dans un système contextuellement observable, on dispose de l'instrumentation nécessaire pour déterminer à l'instant du diagnostic quel

est ce contexte (on voit le sens des échanges avec l'environnement et donc la causalité interne au système). Cette observabilité contextuelle est faible dès lors que tous les paramètres d'entrée sont observés mais pas forcément tous les paramètres de sortie. Si par contre tous les paramètres du contexte sont observés, l'observabilité contextuelle sera forte. Dans cette théorie, les observations étant issues de mesures réelles, elles sont considérées comme une théorie ontologiquement vraie. Cette considération ne restreint pas au système dont l'observation est certaine. En effet, si les observations sont incertaines au sens de [Chen and Patton, 1999],[Adrot et al., 1999], [Patton et al., 2000], [Lamperti and Zanella, 2002], c'est que l'on a modélisé le capteur qui les a produites avec de l'incertitude. La mesure réelle est certaine mais c'est la façon dont elle a été produite qui peut ne pas l'être.

La troisième notion qui est un prérequis au bon fonctionnement d'un système de diagnostic a déjà été étudiée dans la section 2.3.2, c'est la diagnosticabilité que l'on reprend ici avec un autre angle de vue.

Définition 2.36 (Problème diagnosticable) *Le problème* $\mathsf{PD} = (\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS})$ *est diagnosticable s'il existe un unique* $\Delta \in \mathsf{COMPS}$ *tel que*

1. $\sigma(\Delta)$ *est un diagnostic candidat ;*
2. *le bloc diagnosticable* $\mathsf{BD(OBS)}$ *est tel que* $\mathsf{BD(OBS)} \models \sigma(\Delta)$.

La définition précédente est relative au problème (pour une observation OBS donnée) et implique que le résultat de ce problème est certain (pas d'ambiguïté). Si maintenant, cette propriété est vérifiée pour toute observation OBS possible du système alors chaque état de santé possible $\sigma(\Delta)$ est associé à un ensemble de blocs diagnosticables BD(OBS) qui lui sont propres. Soit maintenant φ_Δ telle que

$$\varphi_\Delta \equiv \bigvee_{\mathsf{BD(OBS)} \models \sigma(\Delta)} \mathsf{BD(OBS)},$$

la propriété φ_Δ est diagnosticable selon la définition 2.21 or par définition de φ_Δ, on a bien que $\varphi_\Delta \models \sigma(\Delta)$.

2.4.3.4 Prérequis sur les algorithmes de diagnostic

Les prérequis précédents s'appuient uniquement sur la connaissance du système et sur la nature de ses observations constituant ainsi des problèmes de diagnostic. Tous ces prérequis sont indépendants de l'algorithme de diagnostic qui va effectivement résoudre le problème de diagnostic. Mais il est également possible de formaliser des sources de défaillances d'un système de diagnostic sur les algorithmes qui sont indépendantes de la qualité des modèles et des observations.

Définition 2.37 (Algorithme correct) *Un algorithme de diagnostic* ALG *sur un problème* PD *est correct si* $\mathsf{PD} \vdash_{\mathsf{ALG}} \sigma(\Delta)$ *implique que* $\sigma(\Delta)$ *est un candidat effectif de* PD.

Cette première propriété assure ainsi que l'algorithme, pour un problème PD donné, va déterminer un ensemble de ALG-candidats qui sont tous des candidats effectifs de PD. Si ALG est correct, il va donc en général déterminer un sous-ensemble des candidats effectifs de PD.

Définition 2.38 (Algorithme complet) *Un algorithme de diagnostic* ALG *est* complet *si pour tout candidat de* PD*, on a :*

$$\mathsf{PD} \vdash_{\mathsf{ALG}} \sigma(\Delta).$$

Cette deuxième propriété est duale de la première. Elle garantit que tous les candidats de PD sont effectivement déterminés par ALG. Si ALG est complet, il va donc en général être moins précis (plus ambigu) car il pourra contenir plus de ALG-candidats que de candidats effectifs de PD.

2.4.4 Indicateurs de défaillance d'un système de diagnostic

La sous-section précédente a listé un ensemble de prérequis, de critères de qualité sur un système de diagnostic. Ces prérequis, mis en commun, vont caractériser un ensemble d'indicateurs de défaillance du système de diagnostic. Les résultats présentés sont décrits de façon positive (sous la forme $A \Longrightarrow B$) comme des prérequis au bon fonctionnement du système de diagnostic, l'indicateur associé étant sa contraposée ($\neg B \Longrightarrow \neg A$).

Le premier d'entre eux est trivial mais il est important car il fixe les hypothèses du diagnostic à base de modèle.

Théorème 2.5 ([8]) *Si* DS *est ontologiquement vrai alors* DS *est satisfaisable.*

Si le modèle DS n'est pas satisfaisable alors on sait qu'il y a nécessairement des erreurs de modélisation. [2] L'intérêt de cette propriété est qu'elle est simple à vérifier par un simple test de satisfaisabilité. Le deuxième résultat lie la validité du diagnostic résultat et la vérité ontologique de la connaissance sur laquelle se fonde le système de diagnostic.

Théorème 2.6 ([8]) *Si* DS *et* OBS *sont ontologiquement vrais alors :*

2. En toute rigueur ici, on fait l'hypothèse de l'existence d'un modèle ontologiquement vrai sur une réalité. C'est l'hypothèse du scientifique, mais cette question peut être débattue sur le plan philosophique.

1. *tout système de diagnostic mettant en œuvre un diagnostic de cohérence est valide ;*
2. *un système de diagnostic abductif peut être invalide.*

Le premier point du résultat montre la validité d'un diagnostic de cohérence dès lors que l'on a la vérité ontologique du modèle du système. Ce résultat exprime le pari du diagnostic à base de modèle. La validité de l'approche est reportée sur la qualité du modèle utilisé et non sur l'algorithme qui est un moteur générique de diagnostic de cohérence. Le deuxième point est plus surprenant. Même si DS et OBS sont ontologiquement vrais, il est possible qu'un diagnostic abductif soit invalide. La raison de ce résultat se trouve dans le fait que dans une théorie $\mathsf{DS} \wedge \mathsf{OBS}$ ontologiquement vraie, on n'a pas l'assurance qu'une propriété φ observable soit telle que $\mathsf{DS} \wedge \mathsf{OBS} \wedge \sigma(\Delta) \models \varphi$ ou telle que $\mathsf{DS} \wedge \mathsf{OBS} \wedge \sigma(\Delta) \models \neg\varphi$. La validité du diagnostic abductif impose des conditions plus restrictives sur la théorie $\mathsf{DS} \wedge \mathsf{OBS}$ liées à la complétude.

Théorème 2.7 ([5]) *Soit* DS *un modèle complet, soit* PD *un problème de diagnostic sur* DS *avec une observabilité contextuelle faible dont le contexte observé* CXT *est* (P_E, P_S)*, soit* $\mathsf{OBS_E}$ *les observations de* P_E *et* $\mathsf{OBS_S}$ *les observations potentielles de* P_S *alors pour tout état de santé* $\sigma(\Delta)$

$$\mathsf{DS} \wedge \mathsf{CXT} \wedge \mathsf{OBS_E} \wedge \mathsf{OBS_S} \wedge \sigma(\Delta) \text{ est satisfaisable}$$

ssi

$$\mathsf{DS} \wedge \mathsf{CXT} \wedge \mathsf{OBS_E} \wedge \sigma(\Delta) \text{ est satisfaisable}$$

et

$$\mathsf{DS} \wedge \mathsf{CXT} \wedge \mathsf{OBS_E} \wedge \sigma(\Delta) \models \mathsf{OBS_S}.$$

Ce théorème est le deuxième prérequis pour que le diagnostic abductif soit valide. Il impose en effet que DS soit complet et donc déterministe pour l'ensemble des paramètres d'entrée définis par le contexte et que ces paramètres d'entrée soient observés. La satisfaisabilité du problème devient alors équivalente à l'explication des potentielles observations de $\mathsf{OBS_S}$. La validité du raisonnement abductif est alors assurée si DS et OBS sont de plus ontologiquement vrais. C'est ce qu'énonce le résultat suivant.

Corollaire 2.1 ([5]) *Soit* DS *un modèle complet, soit* PD *un problème de diagnostic sur* DS *avec une observabilité contextuelle forte dont le contexte observé* CXT *est* (P_E, P_S)*, soit* $\mathsf{OBS_E}$ *les observations de* P_E *et* $\mathsf{OBS_S}$ *les observations de* P_S*, si* DS *et* OBS *sont ontologiquement vrais alors le système de diagnostic* $(\mathsf{DS}, \mathsf{COMPS}, \mathsf{CXT} \wedge \mathsf{OBS_E}, \mathsf{OBS_S}, \mathsf{ALG})$ *est valide si* ALG *est un algorithme de diagnostic abductif.*

Les résultats précédents caractérisent des indicateurs de défaillance sur le problème et son observation indépendamment de l'algorithme ALG fourni par le système de diagnostic. Les indicateurs associés à ALG sont les suivants.

Théorème 2.8 ([8]) *Si* ALG *est un algorithme complet et correct pour un problème* PD *alors le diagnostic* D *de* PD *est tel que*

$$\mathsf{PD} \vdash_{\mathsf{ALG}} \mathsf{D}.$$

Pour qu'un algorithme réponde correctement au problème il doit mettre en œuvre deux propriétés complémentaires. La correction garantit que tout candidat généré par ALG est effectivement un candidat du problème. Mais la correction ne garantit pas que toute solution du problème sera déterminée. À l'opposé, la complétude assure l'exhaustivité des solutions de PD mais ne garantit pas qu'elle ne génère pas des candidats non solution de PD. Les notions de correction et de complétude sont issues des notions classiques sur la théorie logique (voir définitions 2.37 et 2.38) mais les effets sur le résultat sont paradoxalement inversés. Un algorithme correct au sens de la définition 2.37 peut retourner un résultat invalide (tous les candidats retournés sont possibles mais le candidat réel peut ne pas avoir été trouvé, typiquement générateur de *faux négatifs*) et un algorithme complet au sens de la définition 2.38 retournera toujours une sur-approximation du résultat attendu (il conserve la validité du résultat mais est moins précis, typiquement générateur de *faux positifs*).

2.4.5 Formalisation et résolution d'un problème de méta-diagnostic

À l'aide des propriétés présentées dans la section précédente, il est désormais possible de formuler et de résoudre automatiquement des problèmes de méta-diagnostic. Un tel problème repose sur la représentation d'un méta-système.

Définition 2.39 (Méta-système [8]) *Un méta-système est un couple* (M-DS, M-COMPS) *où :*

- M-DS *est un ensemble de phrases du premier ordre décrivant un système de diagnostic ;*
- M-COMPS *est un ensemble de constantes représentant les* méta-composants *du système de diagnostic.*

Un méta-composant est une caractéristique du système de diagnostic analysé qui a été initialement identifiée. Au lieu d'identifier un composant réel du système sous-jacent, le méta-composant identifie des parties du système de diagnostic (à savoir des parties de modèles de DS ou des propriétés sur OBS et ALG). Comme pour le choix des composants dans un problème de diagnostic (voir la section 2.2.5), le choix des méta-composants n'est pas unique et dépend du niveau d'abstraction souhaité. M-DS décrit des comportements normaux et anormaux sur le même principe que DS mais en utilisant un prédicat de méta-anomalie M-An() (voir les exemples de la section 2.4.6).

Définition 2.40 (Méta-observations [8]) *Les méta-observations* M-OBS *sont les observations du système* OBS *augmentées des observations sur le système de diagnostic.* M-OBS *est un ensemble de phrases du premier ordre.*

Les méta-observations M-OBS rassemblent les observations OBS du problème traité par le système de diagnostic auxquelles on adjoint des informations observées propres au système de diagnostic. Ces informations sont en général issues d'une expertise externe sur la qualité du diagnostic obtenu. M-OBS peut en effet contenir l'état de santé réel (partiel ou non) qui a pu être déterminé par les agents réparateurs du système qui ont dû effectivement intervenir pour réparer le système suite au résultat de diagnostic fourni par le système de diagnostic (peut-être que les agents réparateurs ont pu confirmer/infirmer que le résultat donné est valide ou non). Bien évidemment, on conserve la distinction entre les méta-observations symptomatiques et contextuelles $\text{M-OBS}_\text{C}, \text{M-OBS}_\text{S}$.

Un problème de méta-diagnostic est défini comme suit.

Définition 2.41 (Problème de méta-diagnostic [8]) *Un problème de méta-diagnostic* M-PD *est un quadruplet* $(\text{M-DS}, \text{M-COMPS}, \text{M-OBS}_\text{C}, \text{M-OBS}_\text{S})$ *où :*

— $(\text{M-DS}, \text{M-COMPS})$ *est la description du méta-système ;*

— $\text{M-OBS} = \text{M-OBS}_\text{C} \wedge \text{M-OBS}_\text{S}$ *sont les méta-observations.*

Dans le cadre du méta-diagnostic, le choix des méta-composants va aussi dépendre des hypothèses posées par le problème. En effet, à l'opposé d'un problème de diagnostic classique, il est nécessaire de poser quelques hypothèses afin de définir un problème de méta-diagnostic donné. Ces hypothèses sont de la forme : *pour ce problème donné, on suppose que* OBS *est ontologiquement vraie* ou bien encore *pour ce problème donné, on suppose que* ALG *est complet.* Si l'on ne fait aucune hypothèse alors très peu d'indicateurs formulés dans la section précédente pourront être exploités. Ces hypothèses sont posées pour un problème donné mais peuvent bien évidemment être remplacées par d'autres pour la définition d'un deuxième problème.

La solution d'un problème de méta-diagnostic est un ensemble de méta-états de santé.

Définition 2.42 (Méta-état de santé [8]) *Soit* $\Phi \subseteq \text{M-COMPS}$*, un méta-état de santé* $\pi(\Phi, \text{M-COMPS} \setminus \Phi)$ *est la conjonction*

$$\pi(\Phi, \text{M-COMPS}) \equiv \bigwedge_{\text{mc} \in \Phi} \text{M-An(mc)} \wedge \bigwedge_{\text{mc} \in \text{M-COMPS} \setminus \Phi} \neg \text{M-An(mc)}.$$

Un méta-état de santé vérifiant les conditions suivantes est une solution au problème de méta-diagnostic.

Définition 2.43 (Méta-candidat [8]) *Un méta-candidat pour le problème de méta-diagnostic* $\mathsf{M\text{-}PD} = (\mathsf{M\text{-}DS}, \mathsf{M\text{-}COMPS}, \mathsf{M\text{-}OBS_C}, \mathsf{M\text{-}OBS_S})$ *est un méta-état de santé* $\pi(\Phi, \mathsf{M\text{-}COMPS})$ *tel que :*

— $\mathsf{M\text{-}DS} \wedge \mathsf{M\text{-}OBS_C} \wedge \pi(\Phi, \mathsf{M\text{-}COMPS})$ *est satisfaisable ;*

— $\mathsf{M\text{-}DS} \wedge \mathsf{M\text{-}OBS_C} \wedge \pi(\Phi, \mathsf{M\text{-}COMPS}) \models \mathsf{M\text{-}OBS_S}$.

Définition 2.44 (Méta-diagnostic [8]) *Le méta-diagnostic du problème* M-PD *est l'ensemble des méta-candidats de* M-PD.

2.4.6 Exemples

Afin d'illustrer la théorie du méta-diagnostic, deux problèmes de diagnostic sont décrits.

2.4.6.1 Le problème de la description DS ontologiquement fausse

En reprenant la figure 2.1, on va supposer à titre illustratif que la description DS du circuit est donnée par la théorie de la figure 2.5. À la différence de la figure 2.2, cette description remplace la phrase $\mathsf{M_1desc}$ par un comportement erroné du multiplieur $\mathsf{M_1}$.

$$
\begin{array}{ll}
\mathsf{M_1desc} : & \neg\mathsf{An}(\mathsf{M_1}) \Longrightarrow (\mathsf{v(x)} = (\mathsf{v(a)} + 1) * \mathsf{v(c)}) \\
\mathsf{M_2desc} : & \neg\mathsf{An}(\mathsf{M_2}) \Longrightarrow (\mathsf{v(y)} = \mathsf{v(b)} * \mathsf{v(d)}) \\
\mathsf{M_3desc} : & \neg\mathsf{An}(\mathsf{M_3}) \Longrightarrow (\mathsf{v(z)} = \mathsf{v(c)} * \mathsf{v(e)}) \\
\mathsf{A_1desc} : & \neg\mathsf{An}(\mathsf{A_1}) \Longrightarrow (\mathsf{v(f)} = \mathsf{v(x)} + \mathsf{v(y)}) \\
\mathsf{A_2desc} : & \neg\mathsf{An}(\mathsf{A_2}) \Longrightarrow (\mathsf{v(g)} = \mathsf{v(y)} + \mathsf{v(z)})
\end{array}
$$

FIGURE 2.5 – Description $\mathsf{DS_{of}}$ ontologiquement fausse du système de la figure 2.1.

On va supposer maintenant que l'on observe un système réel dont la description est donnée par la figure 2.5 et que les observations contextuelles en question sont :

$$\mathsf{OBS_1} \equiv \mathsf{v(a)} = 1 \wedge \mathsf{v(b)} = 2 \wedge \mathsf{v(c)} = 3 \wedge \mathsf{v(d)} = 4 \wedge \mathsf{v(e)} = 5 \wedge \mathsf{v(f)} = 11 \wedge \mathsf{v(g)} = 22$$

et qu'il n'y a pas d'observations symptomatiques. Supposons maintenant que le système de diagnostic utilise l'algorithme DITO , on a donc le système de diagnostic suivant :

$$(\mathsf{DS_{of}}, \mathsf{COMPS}, \mathsf{OBS_1}, \top, \textsc{Dito})$$

où COMPS est l'ensemble des composants $\mathsf{M_1}, \mathsf{M_2}, \mathsf{M_3}, \mathsf{A_1}, \mathsf{A_2}$. Dans ces circonstances, DITO retourne le résultat décrit dans la table 2.2.

M_2, M_3	M_1, M_3	M_1, M_2	A_2, M_2
A_2, M_1	A_1, M_3	A_1, M_2	A_1, A_2

TABLE 2.2 – Candidats minimaux retournés par le système de diagnostic $\mathsf{DiagSys}_1 = (\mathsf{DS}_{of}, \mathsf{COMPS}, \mathsf{OBS}_1, \textsc{Dito})$.

Supposons maintenant qu'après le retour usine, le réparateur informe que selon ses propres tests l'état de santé du système est :

$$\sigma_T \equiv \neg An(M_1) \wedge \neg An(M_2) \wedge \neg An(M_3) \wedge \neg An(A_1) \wedge An(A_2).$$

On en déduit immédiatement que le système de diagnostic $\mathsf{DiagSys}_1$ n'est pas valide. On peut alors définir un problème de méta-diagnostic sur $\mathsf{DiagSys}_1$, il est présenté sur la table 2.3. Dans ce problème de méta-diagnostic, l'algorithme est parfaitement

$\mathsf{M\text{-}COMPS}_1$:	$\{M_1desc, M_2desc, M_3desc, A_1desc, A_2desc\}$
$\mathsf{M\text{-}DS}_1$:	$\neg M\text{-}An(M_1desc) \Longrightarrow (\neg An(M_1) \Longrightarrow (v(x) = (v(a) + 1) * v(c)))$ $\neg M\text{-}An(M_2desc) \Longrightarrow (\neg An(M_2) \Longrightarrow (v(y) = v(b) * v(d)))$ $\neg M\text{-}An(M_3desc) \Longrightarrow (\neg An(M_3) \Longrightarrow (v(z) = v(c) * v(e)))$ $\neg M\text{-}An(A_1desc) \Longrightarrow (\neg An(A_1) \Longrightarrow (v(f) = v(x) + v(y)))$ $\neg M\text{-}An(A_2desc) \Longrightarrow (\neg An(A_2) \Longrightarrow (v(g) = v(y) + v(z)))$
$\mathsf{M\text{-}OBS}_1$:	$v(a) = 1 \wedge v(b) = 2 \wedge v(c) = 3 \wedge v(d) = 4 \wedge v(e) = 5 \wedge v(f) = 11 \wedge$ $v(g) = 22 \wedge \neg An(M_1) \wedge \neg An(M_2) \wedge \neg An(M_3) \wedge \neg An(A_1) \wedge An(A_2)$

TABLE 2.3 – Problème de méta-diagnostic ($\mathsf{M\text{-}DS}_1, \mathsf{M\text{-}COMPS}_1, \mathsf{M\text{-}OBS}_1$)

connu et maîtrisé : il est démontré que DITO est correct et complet. Il faut maintenant fixer une autre dimension du problème. Dans ce problème, on impose que les

observations sont ontologiquement vraies et que le retour usine est ontologiquement vrai. À partir de ces hypothèses et du théorème 2.6, cela indique que le problème se trouve dans $\mathsf{DS}_{\mathsf{of}}$. On se propose donc d'identifier les raisons possibles du dysfonctionnement de $\mathsf{DiagSys}_1$ en définissant un méta-composant comme étant une phrase de $\mathsf{DS}_{\mathsf{of}}$. Le méta-composant $\mathsf{M}_1\mathsf{desc}$ de $\mathsf{M\text{-}COMPS}_1$ est ainsi un symbole associé à la phrase $\neg\mathsf{An}(\mathsf{M}_1) \Longrightarrow (\mathsf{v}(\mathsf{x}) = (\mathsf{v}(\mathsf{a}) + 1) * \mathsf{v}(\mathsf{c}))$ et ainsi de suite. Le méta-système décrit alors le comportement normal de $\mathsf{DiagSys}_1$ associé à ces méta-composants : intuitivement, si le méta-composant $\mathsf{M}_1\mathsf{desc}$ fonctionne correctement alors c'est que la phrase associée $(\neg\mathsf{An}(\mathsf{M}_1) \Longrightarrow (\mathsf{v}(\mathsf{x}) = (\mathsf{v}(\mathsf{a}) + 1) * \mathsf{v}(\mathsf{c}))$ est effectivement vraie d'où la première phrase du méta-système :

$$\neg\mathsf{M\text{-}An}(\mathsf{M}_1\mathsf{desc}) \Longrightarrow (\neg\mathsf{An}(\mathsf{M}_1) \Longrightarrow (\mathsf{v}(\mathsf{x}) = (\mathsf{v}(\mathsf{a}) + 1) * \mathsf{v}(\mathsf{c}))).$$

Concernant les méta-observations $\mathsf{M\text{-}OBS}_1$, elles regroupent les observations du problème de diagnostic initial ainsi que l'observation de l'état de santé réel $\sigma_\top$. Le méta-diagnostic résultat de ce problème est le suivant :

$$\{\mathsf{M\text{-}An}(\mathsf{M}_1\mathsf{desc}), \mathsf{M\text{-}An}(\mathsf{M}_2\mathsf{desc}), \mathsf{M\text{-}An}(\mathsf{A}_1\mathsf{desc})\}.$$

Il confirme que $\mathsf{DS}_{\mathsf{of}}$ est bien ontologiquement fausse mais en plus il suspecte trois phrases parmi cinq comme candidat minimal. On peut poursuivre l'investigation, en s'appuyant sur d'autres problèmes de diagnostic (issus d'observations différentes sur un même système ou bien alors issus de systèmes identiques). Par exemple, supposons que l'on dispose d'un circuit identique mais considéré comme normal (parce qu'il est neuf par exemple) et que sur ce deuxième système sur banc d'essai, on observe

$$\mathsf{OBS}_2 \equiv \mathsf{v}(\mathsf{a}) = 1 \wedge \mathsf{v}(\mathsf{b}) = 2 \wedge \mathsf{v}(\mathsf{c}) = 3 \wedge \mathsf{v}(\mathsf{x}) = 3 \wedge \mathsf{v}(\mathsf{d}) = 4 \wedge \mathsf{v}(\mathsf{e}) = 5 \wedge \mathsf{v}(\mathsf{f}) = 11 \wedge \mathsf{v}(\mathsf{g}) = 23,$$

on peut ainsi définir comme précédemment un deuxième problème de méta-diagnostic en s'appuyant sur le système de diagnostic $\mathsf{DiagSys}_2 = (\mathsf{DS}_{\mathsf{of}}, \mathsf{COMPS}, \mathsf{OBS}_2, \textsc{Dito})$ qui lui va conduire au seul candidat minimal $\mathsf{M\text{-}An}(\mathsf{M}_1\mathsf{desc})$ et qui raffine donc le résultat du premier problème de méta-diagnostic.

2.4.6.2 Le problème de l'algorithme incomplet

Dans cet exemple, la description du système considéré est celle de la figure 2.2. Le premier problème de diagnostic considéré est celui où les observations sont les suivantes :

$$\mathsf{OBS}_3 \equiv \mathsf{v}(\mathsf{a}) = 1 \wedge \mathsf{v}(\mathsf{b}) = 2 \wedge \mathsf{v}(\mathsf{c}) = 3 \wedge \mathsf{v}(\mathsf{d}) = 4 \wedge \mathsf{v}(\mathsf{e}) = 5 \wedge \mathsf{v}(\mathsf{f}) = 11 \wedge \mathsf{v}(\mathsf{g}) = 24.$$

Pour résoudre ce problème, l'algorithme de diagnostic ALG utilisé est un algorithme de recherche de candidat minimal. Le diagnostic résultant du système de diagnostic $\mathsf{DiagSys_3} = (\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS_3}, \mathsf{ALG})$ est le suivant :

1. $\sigma_1 = \mathsf{An}(\mathsf{A_2}) \wedge \bigwedge_{c \in \mathsf{COMPS} \setminus \{\mathsf{A_2}\}} \neg \mathsf{An}(c)$
2. $\sigma_2 = \mathsf{An}(\mathsf{M_3}) \wedge \bigwedge_{c \in \mathsf{COMPS} \setminus \{\mathsf{M_3}\}} \neg \mathsf{An}(c)$.

Après réparation, il s'avère que l'état de santé réel est :

$$\sigma_T = \mathsf{An}(\mathsf{A_1}) \wedge \mathsf{An}(\mathsf{M_2}) \wedge \bigwedge_{c \in \mathsf{COMPS} \setminus \{\mathsf{A_1}, \mathsf{M_2}\}} \neg \mathsf{An}(c)$$

On en déduit immédiatement que $\mathsf{DiagSys_3}$ est invalide. En faisant l'hypothèse que DS et $\mathsf{OBS_3}$ sont ontologiquement vraies, on peut s'appuyer sur les théorèmes 2.6 et 2.8 pour suspecter un problème sur l'algorithme ALG. Deux types potentiels de problèmes sont identifiés sur ALG, la correction ou la complétude.

On reprend $\mathsf{DiagSys_3}$ mais au lieu de faire tourner ALG, on le remplace par Dito et le résultat est le suivant :

1. $\sigma_1 = \mathsf{An}(\mathsf{A_2}) \wedge \bigwedge_{c \in \mathsf{COMPS} \setminus \{\mathsf{A_2}\}} \neg \mathsf{An}(c)$
2. $\sigma_2 = \mathsf{An}(\mathsf{M_3}) \wedge \bigwedge_{c \in \mathsf{COMPS} \setminus \{\mathsf{M_3}\}} \neg \mathsf{An}(c)$
3. $\sigma_3 = \mathsf{An}(\mathsf{M_1}) \wedge \mathsf{An}(\mathsf{M_2}) \wedge \bigwedge_{c \in \mathsf{COMPS} \setminus \{\mathsf{M_1}, \mathsf{M_2}\}} \neg \mathsf{An}(c)$
4. $\sigma_4 = \mathsf{An}(\mathsf{A_1}) \wedge \mathsf{An}(\mathsf{M_2}) \wedge \bigwedge_{c \in \mathsf{COMPS} \setminus \{\mathsf{A_1}, \mathsf{M_2}\}} \neg \mathsf{An}(c)$.

Sachant que Dito est complet et correct, le théorème 2.8 affirme que le diagnostic du problème $(\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS_3})$ est l'ensemble des candidats définis par les candidats minimaux $\sigma_{1\ldots4}$. La description du méta-problème de diagnostic considéré est décrit dans la table 2.4. On définit deux méta-composants sur ALG, à savoir sa complétude et sa correction. Un algorithme est complet s'il retourne au moins tous les candidats de $(\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS_3})$ et il est correct si tous les candidats retournés sont des candidats de $(\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS_3})$. Les méta-observations $\mathsf{M\text{-}OBS_2}$ rassemblent les résultats de Dito et de ALG sur le même problème (tout candidat est un sur-ensemble d'un candidat minimal) sachant que Dito retourne le diagnostic de $(\mathsf{DS}, \mathsf{COMPS}, \mathsf{OBS_3})$.

La résolution de ce problème conduit à déterminer que :

$$\mathsf{M\text{-}An}(\mathsf{ALGCOMPLET})$$

qui signifie que ALG est incomplet et c'est la raison pour laquelle $\mathsf{DiagSys_3}$ est invalide. Il est à noter ici que cet exemple est illustratif, dès lors que ALG ne calcule effectivement que les candidats à cardinalité minimale, il est de facto incomplet mais correct. En pratique l'algorithme utilisé peut être complexe à analyser concernant sa correction et sa complétude et le méta-diagnostic offre un moyen automatique de cerner les limites d'un tel algorithme au cours de son utilisation.

M-COMPS$_2$:
$\{\text{ALGCOMPLET}, \text{ALGCORRECT}\}$
M-DS$_2$:
$\neg\text{M-An}(\text{ALGCOMPLET}) \Longrightarrow (\forall x, \text{PD}(x) \Longrightarrow \text{ALG}(x))$
$\neg\text{M-An}(\text{ALGCORRECT}) \Longrightarrow (\forall x, \text{ALG}(x) \Longrightarrow \text{PD}(x))$
M-OBS$_2$:
$\forall x \in \text{SURENSEMBLE}(A_2), \text{PD}(x \wedge \text{ALG}(x))$
$\forall x \in \text{SURENSEMBLE}(M_3), \text{PD}(x) \wedge \text{ALG}(x))$
$\forall x \in \text{SURENSEMBLE}(M_1, M_2), \text{PD}(x)$
$\forall x \in \text{SURENSEMBLE}(A_1, M_2), \text{PD}(x)$

TABLE 2.4 – Problème de méta-diagnostic de l'algorithme ALG.

2.4.7 Extensions

Le problème décrit précédemment résout des problèmes de méta-diagnostic sur des problèmes de diagnostic statiques tels qu'ils ont été décrits tout au long de ce chapitre. Néanmoins on peut résoudre des problèmes de méta-diagnostic sur des problèmes de diagnostic qui sortent de ce contexte. Dans [4], une extension du méta-diagnostic a été proposée dans le cadre de la résolution d'un problème de diagnostic sur un système à dynamique continue [Isermann and Ballé, 1997] en s'appuyant sur une étude comparative sur le diagnostic au sein des communautés IA et automatique [Cordier et al., 2000b] [Cordier et al., 2000a]. On suppose dans ce travail que la méthode de diagnostic repose sur un modèle d'équations à partir desquelles on peut extraire des résidus [Frisk and Nielsen, 2006]. De ces résidus, on peut déduire une matrice de signatures de fautes qui est une connaissance du système purement logique. C'est à partir de cette connaissance que l'on peut appliquer la théorie du méta-diagnostic afin de déterminer la possibilité d'erreurs et leur localisation dans les équations du modèle en confrontant le résultat effectif du diagnostic et la situation de faute réelle (cas typique d'une description ontologiquement fausse telle que dans le premier exemple de la section précédente).

2.5 Résumé

Les recherches sur le raisonnement logique du diagnostic pour les systèmes statiques ont retrouvé un intérêt grandissant avec l'apparition de machines puissantes et le développement d'algorithmes de calcul de satisfaisabilité qui permettent aujourd'hui de mettre en pratique de tels raisonnements automatiques sur des systèmes réalistes [Feldman et al., 2014], [Jannach et al., 2016]. Dans ce contexte, les travaux effectués au cours de mes collaborations successives tournent tous autour du même axe, à savoir l'amélioration de la qualité, de la performance des outils pour résoudre ce type de problème. La première contribution de cette lignée a été le développement d'un outil de diagnostic logique DITO qui profite de l'engouement actuel de la programmation par contraintes et en particulier des dernières avancées dans la résolution de problèmes à satisfaction de contraintes et la mise au point d'une stratégie de diagnostic réaliste, prenant en compte la complexité intrinsèque du problème afin de calculer les candidats minimaux. Le deuxième point abordé est lié à la qualité des modèles et surtout à l'instrumentation des systèmes sous-jacents. Un système diagnosticable au sens des définitions proposées dans cette contribution peut être exploité avec deux vues différentes. Dans la première orientation (la plus classique), la diagnosticabilité répond à la question suivante : peut-on diagnostiquer un dysfonctionnement f particulier avec certitude à l'aide de l'observabilité intrinsèque du système ? La réponse affirmative à cette question nous informe donc qu'un algorithme de diagnostic de type DITO pourra effectivement résoudre avec certitude le problème. Le deuxième point de vue sur la diagnosticabilité est motivé par le fait que les conditions de diagnosticabilité pour un phénomène particulier sont très fortes, on peut donc essayer de répondre à la question (moins classique) : que peut-on diagnostiquer avec certitude dans un système ? Au lieu de se focaliser sur une propriété particulière, on va plutôt chercher à lister ce qui est diagnosticable, ce qui peut être suffisant pour l'aide à la décision qui sous-tend l'activité de diagnostic. La troisième contribution est le méta-diagnostic. Autour de ce concept on désigne l'analyse logique et formelle de tous les facteurs de défaillance d'un système de diagnostic. Outre les problèmes de modélisation de système (qui ne sont pas propres à un problème de diagnostic par ailleurs), on a formellement identifié un ensemble de facteurs de défaillance et automatisé un raisonnement logique pour détecter et isoler ces problèmes de performances/défaillances dans les systèmes de diagnostic. Comme nous l'avons vu, un raisonnement de méta-diagnostic est un raisonnement de diagnostic et donc on pourrait imaginer que toutes les raisons pour lesquelles un système de diagnostic peut ne pas fonctionner puissent s'appliquer à un système de méta-diagnostic. Ceci n'est néanmoins pas vrai car le cœur du raisonnement du méta-diagnostic repose sur trois propriétés essentielles qui garantissent sa validité :

1. l'outil MEDITO qui résout les problèmes de méta-diagnostic ([6]) s'appuie sur DITO qui est un algorithme de diagnostic de cohérence correct et complet.

2. la description du méta-système est ontologiquement vraie car elle repose sur la description du système (le méta-système ne décrit pas le comportement du système mais le comportement de sa description dont on a la connaissance parfaite (une simple copie))
3. la description des méta-observations est ontologiquement vraie (ce que le système de diagnostic a vu est la copie de ce que le système de méta-diagnostic voit), l'observation des états de santé réel reposant uniquement sous l'hypothèse que l'expert est correct.

Chapitre 3

Diagnostic de comportements séquentiels

Dans le chapitre précédent, le diagnostic a été discuté sous une vision statique à savoir que tout comportement du système étudié est instantané : pour une entrée donnée à un instant t il produit instantanément une sortie et le diagnostic consiste à analyser les observations issues de cet instant t pour déterminer l'état de santé du système. Dans la théorie ainsi présentée, il n'est nullement fait mention d'un quelconque aspect temporel. On peut bien évidemment appliquer le raisonnement à plusieurs instants mais sans hypothèse supplémentaire chaque raisonnement est indépendant et il n'est pas possible de synthétiser un diagnostic sur plusieurs instants. Dans la littérature, il existe des travaux sur le diagnostic incrémental de ces systèmes (on parle aussi dans ce cas de diagnostic séquentiel), par exemple [Feldman et al., 2010b], [Siddiqi and Huang, 2011]. Le diagnostic incrémental est un problème où l'on introduit une première notion temporelle sur ces systèmes à savoir que l'état de santé du système est le même à tout instant de diagnostic. Ainsi, il suffit d'appliquer le raisonnement de diagnostic à plusieurs instants t (et donc pour plusieurs réalisations de la fonction du système) pour affiner le diagnostic : l'ambiguïté du diagnostic ne pouvant que diminuer ou à minima être constante. Le diagnostic incrémental, reposant sur cette hypothèse de constance dans le temps, augmente ainsi les capacités de diagnostic (diagnosticabilité) en mettant par exemple en œuvre des techniques dites *de la prochaine mesure* pour minimiser l'ambiguïté au cours du temps et converger vers un candidat de diagnostic certain.

Ce chapitre est consacré à l'étude du raisonnement diagnostic sur des systèmes où l'état évolue au cours du temps par l'occurrence d'événements. On apporte ici un degré de complexité supérieure, à savoir que l'état global du système à un instant t est caractérisé par la séquence d'événements qui ont eu lieu entre l'instant 0 de la mise en opération du système et cet instant t. Ce type de système guidé par les événements est une sous classe des systèmes dits à événements discrets [Cassandras and Lafortune, 2008]. Nous proposons ici une synthèse de nos

contributions sur ce problème.

3.1 Caractérisation générale du problème et formalisations

À l'opposé des travaux sur le diagnostic statique présenté dans le chapitre précédent qui émane exclusivement de la communauté de l'intelligence artificielle, le diagnostic sur les systèmes à événements discrets est une thématique qui est développée dans plusieurs communautés. La communauté d'intelligence artificielle essentiellement incluse dans la communauté DX s'intéresse à l'étude du raisonnement de diagnostic sur les SED en mettant en place des algorithmes de diagnostic efficaces pour le problème [Baroni et al., 1999], [Lamperti and Zanella, 2002], [Zanella and Lamperti, 2004], [41], [Grastien et al., 2007], [Grastien and Cordier, 2007]. La communauté sciences de l'ingénieur étudie ce problème de façon plus systémique [Zaytoon and Lafortune, 2013] en portant l'accent sur la modélisation effective et l'analyse de propriétés de systèmes à événements discrets tels que des systèmes manufacturiers [Roth et al., 2011] [Philippot et al., 2012], des réseaux de communications [Fabre et al., 2004], de transports [Ghazel and Liu, 2016], des protocoles, etc. Les contributions à ce problème présentées ici sont au carrefour entre ces communautés, elles résument toute la richesse de leurs interactions.

3.1.1 Définition du problème général, un point de vue langage

Le problème du diagnostic de système à événements discrets auquel on s'intéresse a été initié par les travaux de [Lin, 1994] et de [Sampath et al., 1996]. Ce problème est décrit ici sous une autre forme en s'appuyant sur la théorie des langages. Le raisonnement logique associé, analogue à celui du chapitre précédent, sera défini dans la section suivante.

Définition 3.1 (Événement) *Un événement est un phénomène physique ou une abstraction d'un phénomène physique qui est de durée nulle et qui caractérise en général un changement d'état du système sur lequel il se produit. Un système est constitué d'un ensemble d'événements noté* Σ.

On considérera par la suite que Σ est un ensemble fini d'événements autrement dit tout événement réel e_r sera toujours associé à une classe d'équivalence e (e représentera e_r dans Σ) et le nombre de classes d'équivalences e dans Σ est fini.

Définition 3.2 (Comportement/Trajectoire) *Un comportement ou une trajectoire du système est une séquence d'événements de Σ dont tout préfixe est lui-même un comportement. La séquence vide ε est un comportement de tout système.*

À cette notion de trajectoire, on peut adjoindre une opération servant souvent à abstraire une trajectoire, c'est la notion de projection, opération qui sera très utile par la suite.

Définition 3.3 (Projection) *Soit Σ et Σ' deux alphabets d'événements, soit τ une séquence d'événements de Σ, la projection $P_{\Sigma'}() : \Sigma^* \longrightarrow \Sigma'^*$ de τ sur Σ' est définie comme suit :*

$$P_{\Sigma'}(\tau) = \begin{cases} \varepsilon & si\ \tau = \varepsilon, \\ e.P_{\Sigma'}(\tau_1) & si\ \tau = e.\tau_1\ et\ e \in \Sigma', \\ P_{\Sigma'}(\tau_1) & si\ \tau = e.\tau_1\ et\ e \notin \Sigma'. \end{cases}$$

La projection $P_{\Sigma'}(\mathcal{L})$ d'un quelconque langage $\mathcal{L}$ est :

$$P_{\Sigma'}(\mathcal{L}) = \{P_{\Sigma'}(\tau) \in \Sigma'^*\ pour\ tout\ \tau \in \mathcal{L}\}.$$

Un système est caractérisé par l'ensemble de ses comportements. À un instant t, nous ferons l'hypothèse que le comportement du système est une séquence finie d'événements (le système ne peut réaliser une infinité d'événements en un temps fini). L'ensemble des comportements forme donc un langage $\mathcal{S}$ inclus dans la fermeture de Kleene de Σ [Hopcroft et al., 2001] : $\mathcal{S} \subseteq \Sigma^*$. Le langage $\mathcal{S}$ est un langage clos par préfixe par définition des comportements (définition 3.2).

Définition 3.4 (Modèle du système SED) *Un système à événements discrets est représenté par son langage de comportements $\mathcal{S} \subseteq \Sigma^*$ qui est un langage régulier clos par préfixe.*

La régularité du langage $\mathcal{S}$ vient du fait qu'il est clos par préfixe. D'un point de vue langage, l'état courant du système est caractérisé par la séquence d'événements qui a eu lieu depuis son initialisation.

Définition 3.5 (État du SED) *Un état de SED est une trajectoire $\tau \in \mathcal{S}$ du système.*

Cette notion d'état du système semble inutile car elle est redondante avec la notion de trajectoire, elle est néanmoins importante à mentionner car elle affirme que l'état d'un SED sur lequel portent nos travaux n'est défini *que* par la séquence d'événements qui mène à cet état. Par définition, le nombre d'états d'un SED est généralement infini (dès lors que $|\mathcal{S}| = \infty$). Néanmoins avec la nature régulière du langage $\mathcal{S}$, il est

possible d'avoir une représentation abstraite et finie de tous les états à l'aide de formalismes tels que les automates, ou les réseaux de Petri, … Chaque état de l'automate, chaque marquage accessible du réseau de Petri étant le regroupement d'un ensemble potentiellement infini d'états du SED sous-jacent.

Comme dans tout problème de diagnostic, il est nécessaire de caractériser les observations du système. Dans le cadre des SED, cela se fait en général à l'aide d'une fonction d'observation, appelée également un masque observable [Jiang et al., 2001], [Lamperti and Zanella, 2003], [Grastien and Cordier, 2007]. On note Σ_o l'ensemble des événements que l'on peut observer sur le système.

Définition 3.6 (Masque observable événementiel) *Un masque observable événementiel est une fonction : $obs : \Sigma \longrightarrow 2^{\Sigma_o \cup \{\varepsilon\}}$ qui associe à chaque événement e de Σ un ensemble possible d'événements observables de Σ_o ou ε si cet événement n'est pas observable.*

La notion de masque observable a été introduite pour des raisons de généralité en s'appuyant sur les faits suivants.

- Un événement e du système peut être totalement observable, autrement dit, il est identifiable par l'observation, ce cas est représenté par $obs(e) = \{e\}$.
- Un événement e du système peut à l'inverse ne pas être du tout observable ce qui est représenté ici par $obs(e) = \{\varepsilon\}$. ε représente la séquence vide dans Σ_o^* à savoir que si le système produit la séquence $e_1.e.e.e.e_2$ et $obs(e_1) = \{e_1\}$, $obs(e_2) = \{e_2\}$ alors la séquence observée est $\sigma = e_1.\varepsilon.\varepsilon.\varepsilon.e_2$ qui est équivalente à $\sigma = e_1.e_2$ (par ε-réduction classique sur Σ_o^*).
- Un événement e peut-être partiellement observable, son observation ne le caractérise pas complètement, ce cas est représenté par $obs(e) = \{e'\}$.
- Un événement e peut avoir une observation incertaine, son occurrence peut résulter en plusieurs manifestations observables ou bien même il peut parfois être totalement inobservable, ce cas est représenté par un ensemble $obs(e) = \{e_1, e_2, e_3, \ldots, \varepsilon\}$.

Le masque observable, défini sur de simples événements, peut s'étendre aux séquences pour obtenir le *masque observable séquentiel* comme suit :

$$obs^* : \Sigma^* \longrightarrow 2^{\Sigma_o^*}.$$

La séquence observable $\sigma = o_1 \ldots o_n$ appartient à $obs^*(\tau)$ pour $\tau = e_1 \ldots e_n \in \Sigma^*$ si on a $\forall i \in \{1, \ldots, n\}, o_i \in obs(e_i)$.

Définition 3.7 (Modèle du système observé) *Le modèle d'un système observé est un triplet $\mathsf{DS} = (\mathcal{S}, \Sigma_o, obs)$ tel que :*

- *$\mathcal{S}$ est le modèle d'un système sur un alphabet Σ ;*
- *Σ_o est un ensemble d'événements observables ;*

— $obs^* : \Sigma^* \longrightarrow 2^{\Sigma_o^*}$ *est un masque observable séquentiel.*

En diagnostic statique, l'objectif est de déterminer quels composants sont anormaux, quels sont leurs modes de fonctionnement. Pour les systèmes à événements discrets, l'objectif porte essentiellement sur l'occurrence possible d'un ensemble d'événements d'intérêt (souvent qualifiés d'événements anormaux, d'événements de faute, d'événements de panne, défaillance) [Sampath et al., 1995].

Définition 3.8 (Événement d'intérêt) *Un événement d'intérêt est un événement qui a lieu sur le système et dont l'occurrence effective sur le système nécessite une décision d'action sur le système.*

Cette définition d'événements d'intérêt se veut volontairement très générale car le raisonnement diagnostic décrit ici est commun à un ensemble important de problèmes. Un événement d'intérêt peut être :

1. un événement de faute, de panne (la manifestation physique sur le système d'un phénomène qui conduit le système à ne plus pouvoir remplir sa fonction, ce qui nécessite réparation)
2. un événement inattendu qui nécessite que le système soit reconfiguré.

Cette notion d'événement d'intérêt est associée à une manifestation instantanée du système. Cette notion d'événement peut être étendue à la notion de motifs d'événements [25].

Définition 3.9 (Motif d'intérêt) *Un motif d'intérêt M sur un alphabet Σ_M est un langage sous-ensemble de Σ_M^* tel que si $\tau \in M$ alors $\forall \tau' \in \Sigma_M^* \setminus \{\varepsilon\}$, $\tau\tau' \notin M$.*

Un motif représente un comportement bien particulier défini comme un agencement d'événements [Jéron et al., 2006], ou encore une spécification [Jiang and Kumar, 2004]. Si le diagnostic détermine que ce motif de comportement a eu lieu dans un système, il est alors possible de prendre une décision (maintenance, réparation, reconfiguration, replanification...). Par exemple, si le nombre n d'occurrences d'un événement e dans le système est inférieur à k, on peut peut-être considérer que le système fonctionne toujours correctement (sous le seuil de fautes intermittentes acceptables) par contre si ce seuil est franchi ($n = k+1$), le comportement est considéré anormal et l'on doit y remédier. Dans ce cas, typiquement le motif d'intérêt est la séquence $\tau = e \ldots e, |\tau| = k+1$. On note $\mathcal{M}(S)$ la sélection des motifs d'intérêt pour un système donné.

Définition 3.10 (État de santé) *Un état de santé Δ du système est un ensemble de motifs d'intérêt : $\Delta \subseteq \mathcal{M}$.*

L'état de santé d'un SED est ainsi caractérisé dans ce cadre par l'ensemble des motifs *qui ont eu lieu* sur le système ce qui est formalisé par la notion de concordance.

Définition 3.11 (μ-concordance) *Soit τ et μ deux séquences finies sur un alphabet Σ, la séquence τ est μ-concordante si*

1. *$\mu = \varepsilon$; ou*
2. *il existe une décomposition $\tau = \tau_1.e_1.\tau_2$, $e_1 \in \Sigma$ tel que $\mu = e_1.\mu_2$ et τ_2 est μ_2-concordante.*

Définition 3.12 (Concordance de motifs) *Soit $\tau \in \mathcal{S}$ un état du système, soit M un motif d'intérêt, on dit que τ est M-concordante s'il existe au moins une séquence $\mu \in M$ telle que τ soit μ-concordante.*

Le problème général du diagnostic dans les SED peut maintenant être posé.

Définition 3.13 (Problème de diagnostic de SED) *Un problème de diagnostic est un triplet* $(\mathsf{DS}, \mathcal{M}, \mathsf{OBS})$ *:*

— DS *est le modèle d'un système observé sur un alphabet Σ et sur un alphabet observable Σ_o ;*
— $\mathcal{M}$ *est un ensemble de motifs d'intérêt sur un alphabet $\Sigma_{\mathcal{M}} \subseteq \Sigma$;*
— OBS *est une séquence d'observations (*$\mathsf{OBS} \in \Sigma_o^*$*).*

La solution à ce problème est l'ensemble des diagnostics candidats.

Définition 3.14 (Diagnostic candidat) *Un diagnostic candidat de* $(\mathsf{DS}, \mathcal{M}, \mathsf{OBS})$ *est un état de santé Δ tel que :*

cas 1 $(\mathsf{OBS} = \varepsilon)$ **:** *Δ est vide ($\Delta = \emptyset$);*

cas 2 $(\mathsf{OBS} \neq \varepsilon)$ **:** *il existe une trajectoire $\tau = \tau'.e \in \mathcal{S}$, $obs(e) \setminus \{\varepsilon\} \neq \emptyset$ telle que :*

1. *$\mathsf{OBS} \in obs^*(\tau)$; et*
2. *$\mathsf{OBS} \notin obs^*(\tau')$;*
3. *$M \in \Delta$ ssi τ est M-concordante.*

Définition 3.15 (Solution du problème) *Le diagnostic du problème* $(\mathsf{DS}, \mathcal{M}, \mathsf{OBS})$ *est l'ensemble des candidats.*

3.1.2 Quelques variantes/extensions du problème initial

La définition 3.14 caractérise la solution du problème de diagnostic comme un ensemble de candidats, chaque candidat indiquant l'ensemble des motifs *qui ont pu avoir lieu*. Ce problème est défini pour une séquence d'observations OBS donnée (problème qui est usuellement classé dans la littérature sous le terme de diagnostic *hors-ligne*). De par la nature dynamique du système, on peut vouloir mettre à profit le diagnostic obtenu pour une séquence OBS_1 afin de calculer incrémentalement le diagnostic pour une séquence $\mathsf{OBS}_1\mathsf{OBS}_2$ (diagnostic incrémental nécessaire par exemple pour du diagnostic *en-ligne*). Dans ce cas, la connaissance des états de santé Δ_1 possibles après observations de OBS_1 ne suffit pas pour reprendre et pour mettre à jour *directement* le diagnostic sur l'observation de OBS_1 suivi de OBS_2. Pour un suivi de système plus efficace, il faut ajouter une information supplémentaire au diagnostic, à savoir un état.

Définition 3.16 (Diagnostic candidat-état) *Un diagnostic candidat-état de* $(\mathsf{DS}, \mathcal{M}, \mathsf{OBS})$ *est un couple* (τ, Δ) *tel que :*

cas 1 $(\mathsf{OBS} = \varepsilon)$ **:** Δ *est vide* ($\Delta = \emptyset$), $\tau = \varepsilon$ *;*

cas 2 $(\mathsf{OBS} \neq \varepsilon)$ **:** $\tau = \tau'.e \in \mathcal{S}, obs(e) \setminus \{\varepsilon\} \neq \emptyset$ *telle que :*

1. $\mathsf{OBS} \in obs^*(\tau)$ *; et*
2. $\mathsf{OBS} \notin obs^*(\tau')$ *;*
3. $M \in \Delta$ *ssi* τ *est* M*-concordante.*

Là encore, cette définition est indépendante d'un quelconque choix de formalisme et repose sur l'état réel d'un SED. Dans la pratique, le SED est représenté par des formalismes où la représentation des états est finie (automate [Sampath et al., 1995], [Grastien et al., 2007], automates communicants [Baroni et al., 1999], [Rozé and Cordier, 2002], [41] et plus récemment pour le diagnostic en réseau de Petri [Lefebvre and Delherm, 2007], [Basile et al., 2009], [Cabasino et al., 2010]) et ainsi le candidat-état est représenté par un couple (x, Δ) où x est la classe d'équivalence de l'état τ (à savoir x peut être un état d'un automate, ou encore un marquage accessible d'un réseau de Petri).

Une deuxième extension au problème initial est de considérer dans le diagnostic la possibilité que le système *ait évolué après le dernier événement observé*. Dans ce cas, on inclut donc dans la solution l'ensemble des candidats issus de la *fermeture silencieuse* [Lamperti and Zanella, 2003]. L'utilisation de la fermeture silencieuse est cruciale notamment dès lors que l'on considère la résolution décentralisée voire distribuée du diagnostic de SED. Cette extension supprime une restriction incluse dans les définitions précédentes ce qui rend sa définition plus simple.

Définition 3.17 (Diagnostic candidat-état fermé) *Un diagnostic candidat-état fermé de* $(\mathsf{DS}, \mathcal{M}, \mathsf{OBS})$ *est un couple* (τ, Δ) *tel que* $\tau \in \mathcal{S}$ *et :*

1. $\mathsf{OBS} \in obs^*(\tau)$*; et*
2. $M \in \Delta$ *ssi* τ *est* M*-concordante.*

3.2 Spectre d'algorithmes de diagnostic symboliques

Cette section présente l'ensemble des contributions réalisées sur la mise au point et le développement d'algorithmes de diagnostic de SED par des méthodes de type symbolique.

3.2.1 Contexte des travaux

Les travaux décrits dans cette section émanent essentiellement de ma collaboration avec Anika Schumann et Sylvie Thiébaux lorsque j'étais en poste à l'université nationale d'Australie à Canberra de 2003 à 2006 (travaux de master et de thèse d'Anika Schumann, travaux que j'ai co-supervisés avec Sylvie Thiébaux). Ces travaux ont également été poursuivis au cours de ma collaboration avec XingYu Su et Alban Grastien de 2012 à 2015 (travaux de thèses co-supervisés avec Alban Grastien et Patrick Haslum de NICTA-Canberra).

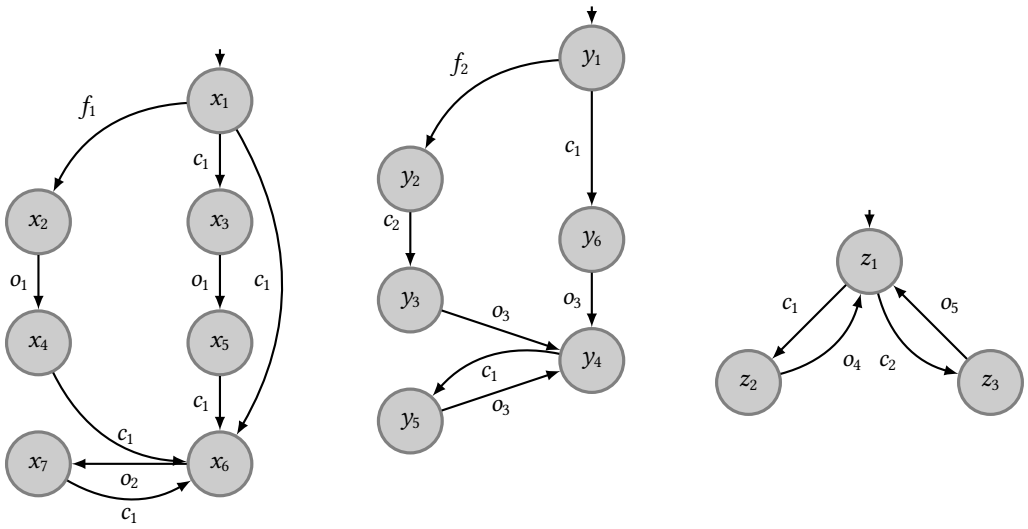
3.2.2 Problématique

3.2.2.1 Modélisation compositionnelle

Le problème du diagnostic de SED présenté dans la section 3.1.1 a été introduit par [Sampath et al., 1995] en s'appuyant sur le formalisme des automates (voir la figure 3.1). Le système est vu comme un ensemble de composants en interaction et le langage du système $\mathcal{S}$ est défini par la synchronisation des langages de composants. Dans ce travail initial, les motifs d'intérêt sont limités à de simples occurrences d'événements fautifs $f \in \Sigma_f$. De même, la notion de masque observable est réduite à la parfaite observation d'un événement du système (un événement e est observable si et seulement si $obs(e) = \{e\}$ et il est non-observable si et seulement si $obs(e) = \{\varepsilon\}$).

Définition 3.18 (Modèle automate de composant) *Le modèle automate d'un composant* C_i *est un quadruplet* $\mathcal{A}_i = (Q_i, \Sigma_i, T_i, q_{0i})$ *où*

1. Q_i *est un ensemble fini d'états;*
2. Σ_i *est un ensemble fini d'événements;*
3. $T_i \subseteq Q_i \times \Sigma_i \times Q_i$ *est un ensemble fini de transitions;*



(a) Automate $\mathcal{A}_1$. (b) Automate $\mathcal{A}_2$. (c) Automate $\mathcal{A}_3$.

FIGURE 3.1 – Un système à événements discrets représenté par 3 automates.

4. *q_{0i} est l'état initial.*

Parmi les événements de Σ_i, on distingue donc les événements observables $\Sigma_{oi} \subseteq \Sigma_i$ des événements non-observables $\Sigma_{noi} = \Sigma_i \setminus \Sigma_{oi}$. Une transition $(q, e, q') \in T_i$ est notée $q \xrightarrow{e} q'$. Un chemin de transitions de longueur n est noté $q^0 \xrightarrow{e^0} q^1 \xrightarrow{e^1} q^2 \dots q^{n-1} \xrightarrow{e^{n-1}} q^n$. Le langage du composant est ainsi caractérisé par le langage généré par l'automate $\mathcal{A}_i$ [Hopcroft et al., 2001] :

$$\mathcal{L}(C_i) \stackrel{\text{def}}{=} \mathcal{L}(\mathcal{A}_i) = \{\tau = e_1 \dots e_n \in \Sigma_i^* : \exists q_{0_i} \xrightarrow{e_1} q_{1_i} \dots q_{n-1_i} \xrightarrow{e_n} q_{n_i} \text{ dans } \mathcal{A}_i\}$$

Pour des raisons de concision et d'efficacité, tout événement non significatif (c.-à-d. tout événement non fautif, non observable et non impliqué dans l'interaction entre composants) ne sera pas considéré dans la modélisation des automates par la suite. Autrement dit, si $\Sigma^i_{\neg sig}$ est l'ensemble des événements non-significatifs du composant C_i, on considérera alors comme modèle de C_i un automate $\mathcal{A}'_i$ tel que :

$$\mathcal{L}(\mathcal{A}'_i) = P_{\Sigma \setminus \Sigma^i_{\neg sig}}(\mathcal{L}(C_i)).$$

La figure 3.1 représente les automates d'un système à trois composants. Dans cette illustration, les événements observables sont $\Sigma_{o_1} = \{o_1, o_2\}$, $\Sigma_{o_2} = \{o_3\}$, $\Sigma_{o_3} = \{o_4, o_5\}$. Les événements fautifs sont f_1 (qui se produit sur le composant C_1) et f_2 (qui se produit sur le composant C_2). À titre d'exemple, le langage $\mathcal{L}(C_3)$ est caractérisé par l'automate

$\mathcal{A}_3$ de la façon suivante. On considère l'état z_1, tout chemin menant de z_1 à z_1 représente un comportement possible de C_1, en utilisant la notation des expressions rationnelles, l'ensemble des comportements possibles de C_3 menant à z_1 est :

$$\mathcal{L}(\mathcal{A}_3, z_1) = (c_1.o_4 + c_2.o_5)^*,$$

de même, on a

$$\mathcal{L}(\mathcal{A}_3, z_2) = c_1.(o_4.(c_2.o_5)^*.c_1)^*,$$
$$\mathcal{L}(\mathcal{A}_3, z_3) = c_2.(o_5.(c_1.o_4)^*.c_2)^*,$$

et finalement

$$\mathcal{L}(C_3) = \mathcal{L}(\mathcal{A}_3) = \mathcal{L}(\mathcal{A}_3, z_1) + \mathcal{L}(\mathcal{A}_3, z_2) + \mathcal{L}(\mathcal{A}_3, z_3).$$

Pour représenter l'interaction entre les composants, on utilise un produit synchronisé d'automates [Arnold, 1992]. Dans le composant C_i on distingue parmi les événements de Σ_i un ensemble d'événements de synchronisation (on parle également d'événements partagés, d'événements interactifs ou de communication). Tout événement de synchronisation de Σ_i fait également parti d'au moins un ensemble d'événements $\Sigma_j, j \neq i$ d'un autre composant C_j. Dans l'exemple de la figure 3.1, il s'agit des événements c_1 et c_2.

Définition 3.19 (Produit synchronisé d'automates) *Le produit synchronisé de deux automates* $\mathcal{A}_i = (Q_i, \Sigma_i, T_i, q_{0i})$ *et* $\mathcal{A}_j = (Q_j, \Sigma_j, T_j, q_{0j})$ *est l'automate* $\mathcal{A}_i||\mathcal{A}_j = (Q_i \times Q_j, \Sigma_i \cup \Sigma_j, T_i||T_j, (q_{0i}, q_{oj}))$ *où* $(q_1, q_2) \xrightarrow{e} (q_1', q_2') \in T_i||T_j$ *ssi :*

1. $q_1 \xrightarrow{e} q_1' \in T_i, q_2' = q_2$ *et* $e \notin \Sigma_i \cap \Sigma_j$ *; ou*
2. $q_2 \xrightarrow{e} q_2' \in T_j, q_1' = q_1$ *et* $e \notin \Sigma_i \cap \Sigma_j$ *; ou*
3. $q_1 \xrightarrow{e} q_1' \in T_i$ *et* $q_2 \xrightarrow{e} q_2' \in T_j$.

Le produit synchronisé s'étend naturellement à n automates de plus il est commutatif ($\mathcal{A}_i||\mathcal{A}_j = \mathcal{A}_j||\mathcal{A}_i$ aux permutations près). Le modèle du système de n composants (dit aussi le modèle global) est ainsi obtenu par synchronisation :

$$\mathcal{A} = \mathcal{A}_1|| \ldots ||\mathcal{A}_n$$

et le langage du système associé est :

$$\mathcal{S} \stackrel{\text{def}}{=} \mathcal{L}(\mathcal{A}).$$

La figure 3.2 présente l'automate $\mathcal{A}_2||\mathcal{A}_3$ résultat du produit synchronisé de $\mathcal{A}_2$ et de $\mathcal{A}_3$ (figure 3.1), les événements de synchronisation entre ces deux composants étant

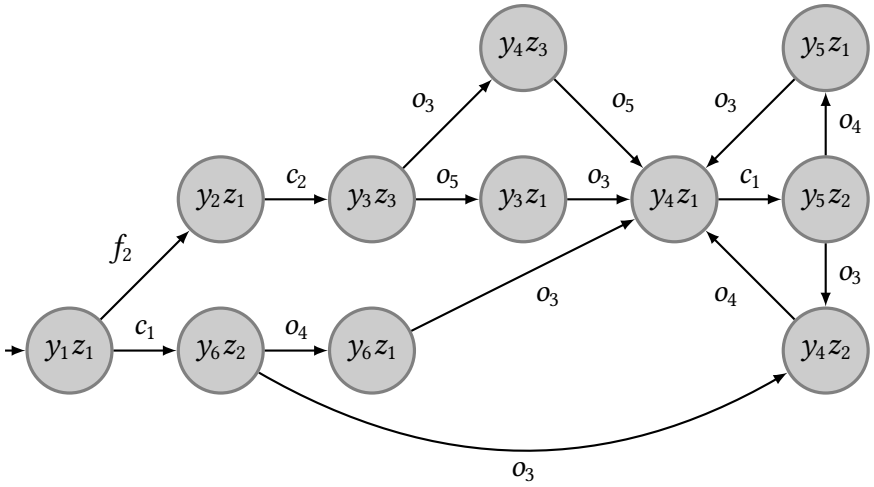


FIGURE 3.2 – Produit synchronisé $\mathcal{A}_2 \| \mathcal{A}_3$.

$\{c_1, c_2\}$. La taille de $\mathcal{A}_2$ est de 6 états et 7 transitions et celle de $\mathcal{A}_3$ est de 3 états et 4 transitions, la taille du produit résultant est de 11 états et 15 transitions. Si à ce produit on intègre également le composant $\mathcal{A}_1$, on se retrouve avec un automate $\mathcal{A}_1 \| \mathcal{A}_2 \| \mathcal{A}_3$ de 42 états et de 78 transitions, ce qui donne un aperçu de la complexité exponentielle au pire du produit de n automates.

3.2.2.2 Problème de diagnostic

Le problème de diagnostic est alors le suivant : $(\mathsf{DS}, \bigcup_{i=1}^{n} \Sigma_{fi}, \mathsf{OBS})$.

- DS est le modèle du système observé représenté par $\mathcal{A}_1 \| \ldots \| \mathcal{A}_n$ et le masque observable *obs* est tel que pour tout événement e de $\bigcup_{i=1}^{n} \Sigma_{oi}$ $obs(e) = \{e\}$ et pour tout autre événement e', $obs(e') = \{\varepsilon\}$.
- $\Sigma_{fi} \subseteq \Sigma_i$ est l'ensemble des événements fautifs (événements d'intérêt) du composant C_i.
- $\mathsf{OBS} \in \left(\bigcup_{i=1}^{n} \Sigma_{oi}\right)^* = \Sigma_o^*$.

La méthode proposée par [Sampath et al., 1995] pour calculer le diagnostic est de construire hors-ligne (sans l'aide des observations OBS) un automate déterministe (communément appelé *automate diagnostiqueur*) qui ne contient que des transitions observables et dont les états contiennent le diagnostic solution du problème si les observations sont caractérisées par un chemin de transitions de l'automate menant à cet état (dans [Sampath et al., 1995], le diagnostic considéré est celui décrit par la définition 3.14). Ensuite, pour une séquence OBS particulière, il suffit de suivre cet automate à partir de son état initial afin d'arriver à l'état solution. Le gros intérêt de cette méthode est que le calcul du diagnostic est efficace, il ne dépend que de la taille de OBS : si un nouvel événement observé est disponible, la mise à jour du diagnostic consiste à

sélectionner la transition de l'automate correspondante, ce qui est une opération en temps constant et immédiate. Malheureusement, la complexité temporelle de la solution est obtenue au détriment de la complexité spatiale. En effet, soit n_c le nombre de composants et n_f le nombre d'événements fautifs, la taille de l'automate $\mathcal{A}$ du modèle global est au pire en $o(2^{n_c})$ par la définition du produit synchronisé. L'automate diagnostiqueur est ensuite obtenu par projection observable puis déterminisation de l'automate $\mathcal{A}$ ce qui implique que sa taille au pire cas est en $o(2^{2^{n_c}})$. À cela s'ajoute que le diagnostiqueur doit gérer n_f fautes possibles et leurs combinaisons (fautes multiples) sa taille est donc également en $o(2^{n_f})$. De ce constat résulte le problème majeur de la résolution d'un problème de diagnostic de SED : la *complexité de son calcul*. La méthode de l'automate diagnostiqueur propose une précompilation du problème qui est maximale (transfert total de la complexité du problème en complexité spatiale) ce qui la rend impraticable même pour un nombre faible de composants.

Les travaux effectués avec Anika Schumann partent de ce constat. L'idée est de définir de nouveaux algorithmes résolvant le même problème en utilisant des techniques dites *symboliques*. Ces techniques portent le nom de *symbolique* dans le sens où les structures de données utilisées symbolisent un ensemble d'informations (des états, des transitions) en les factorisant et en rendant ainsi leur manipulation plus efficace [Darwiche and Marquis, 2002]. La structure de données utilisée a été le diagramme de décision binaire.

3.2.3 Modélisation symbolique du problème

Toute la modélisation et la résolution du problème reposent sur une structure de données : le diagramme de décision binaire [Bryant, 1986] et plus précisément le diagramme de décision binaire ordonné (OBDD : *ordered binary decision diagram*, on utilisera par la suite l'acronyme BDD mais il s'agira toujours d'OBDDs) [Sztipanovits and Misra, 1996], [Somenzi, 1998], [Marchand and Rozé, 2002]. Un BDD est un graphe orienté acyclique muni d'un nœud racine, chaque nœud a deux arcs de sortie (arc haut, arc bas) sauf deux d'entre eux qui n'ont pas d'arcs de sorties (ce sont les nœuds feuilles). Un des nœuds feuilles est associé à la valeur booléenne *vrai* ($\top$) et l'autre à la valeur booléenne *faux* ($\bot$). Quant aux autres nœuds, ils sont associés à des *variables booléennes*, chaque variable étant associée à un nœud et un seul. Cette structure offre un moyen de représenter toute fonction booléenne $(0,1)^n \longrightarrow (0,1)$ de n variables booléennes. La figure 3.3 présente un diagramme de décision binaire pour la formule logique $(\mathsf{a}_1 \wedge \mathsf{b}_1) \vee (\mathsf{a}_2 \wedge \mathsf{b}_2) \vee (\mathsf{a}_3 \wedge \mathsf{b}_3)$. Ce diagramme repose sur l'ordre des variables $\langle \mathsf{a}_1, \mathsf{b}_1, \mathsf{a}_2, \mathsf{b}_2, \mathsf{a}_3, \mathsf{b}_3 \rangle$. Tout chemin de a_1 menant à *Vrai* représente une interprétation logique vraie (un modèle) de la formule. Tout chemin de a_1 menant à *Faux* représente une interprétation logique fausse de la formule. Les arcs pleins symbolisent l'affectation de la variable à vrai du nœud dont l'arc est issu (arcs haut). Les arcs pointillés en symbolisent l'affectation à faux (arcs bas). Par exemple, le chemin

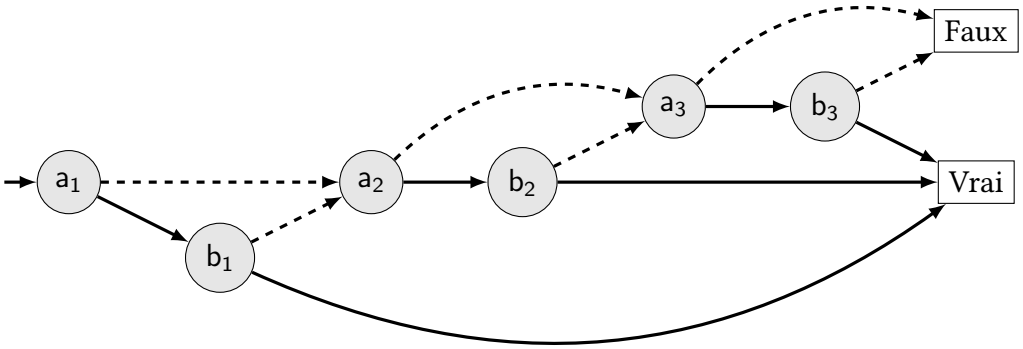


FIGURE 3.3 – Représentation de $(a_1 \wedge b_1) \vee (a_2 \wedge b_2) \vee (a_3 \wedge b_3)$.

$a_1 \longrightarrow a_2 \longrightarrow b_2 \longrightarrow$ Vrai représente que toute interprétation logique telle que a_1 est faux, a_2 est vrai, b_2 est vrai est un modèle de $(a_1 \wedge b_1) \vee (a_2 \wedge b_2) \vee (a_3 \wedge b_3)$, etc.

L'objectif des travaux [63] est de mettre à profit cette structure de données et les opérateurs qui lui sont associés pour développer des algorithmes de diagnostic efficaces. L'intérêt en effet de cette représentation réside dans deux points :

1. une formule logique représente *symboliquement* l'ensemble de ses modèles (ses interprétations logiques vraies), elle offre donc un moyen *non-énumératif, factorisant* de représenter des ensembles et ainsi de réduire la taille mémoire ;
2. l'ensemble des opérateurs $f \star g$ sur les BDD (sur les formules logiques f et g) sont efficaces, notamment l'égalité de deux BDDs est constante, tout autre opérateur a une complexité au pire en $o(|f|.|g|)$ où $|f|$ (respectivement $|g|$) est le nombre de variables impliquées dans la formule f (respectivement g).

La taille effective d'un BDD (qui peut être au pire exponentielle en le nombre de variables booléennes) dépend notamment de l'ordre des variables booléennes, néanmoins il est possible de changer dynamiquement cet ordre à l'aide d'heuristique afin d'obtenir un ordre réduisant la taille du BDD.

Définition 3.20 (Modèle symbolique d'un composant [65]) *La représentation symbolique d'un composant C_i est un septuplet*

$$sC_i = (f^{Q_i}, f'^{Q_i}, f^{\Sigma}, f^{q_{0_i}}, f^{T_{0_i}}, f^{T_{s_i}}, f^{T_{f_i}})$$

avec $\mathcal{A}_i = (Q_i, \Sigma_i, q_{0_i}, T_{o_i} \cup T_{s_i} \cup T_{f_i})$ l'automate de C_i et Σ l'ensemble de tous les événements du système ($\Sigma_i \subseteq \Sigma$).

La formule f^{Σ} représente symboliquement l'ensemble des événements Σ. La raison pour laquelle la formule recense tous les événements et non pas seulement ceux de Σ_i sera expliquée ultérieurement. Soit $\{e_1, \ldots, e_p\} = \Sigma$, pour construire f^{Σ}, on va créer

$l = \lceil log_2(p) \rceil$ variables booléennes $\mathsf{b}_1^\Sigma \dots \mathsf{b}_l^\Sigma$ et chaque événement e_j est associé à une affectation de valeur de vérité à chaque variable et à une seule. Reprenons par exemple l'ensemble des événements de la figure 3.1, $\Sigma = \{f_1, f_2, c_1, c_2, o_1, o_2, o_3, o_4, o_5\}$ alors on utilise 4 variables booléennes $\{\mathsf{b}_1^\Sigma, \mathsf{b}_2^\Sigma, \mathsf{b}_3^\Sigma, \mathsf{b}_4^\Sigma\}$ (pouvant donc coder 16 valeurs de 0 à 15), on peut associer f_1 à la valeur 0 correspondant au modèle de la formule $\mathsf{f}^{\mathsf{f}_1} \equiv \neg\mathsf{b}_1^\Sigma \wedge \neg\mathsf{b}_2^\Sigma \wedge \neg\mathsf{b}_3^\Sigma \wedge \neg\mathsf{b}_4^\Sigma$, f_2 à la valeur 1 ($\mathsf{f}^{\mathsf{f}_2} \equiv \neg\mathsf{b}_1^\Sigma \wedge \neg\mathsf{b}_2^\Sigma \wedge \neg\mathsf{b}_3^\Sigma \wedge \mathsf{b}_4^\Sigma$), et ainsi de suite. L'ensemble Σ est alors représenté par

$$\mathsf{f}^\Sigma \equiv \mathsf{f}^{\mathsf{f}_1} \vee \mathsf{f}^{\mathsf{f}_2} \vee \mathsf{f}^{\mathsf{c}_1} \vee \mathsf{f}^{\mathsf{c}_2} \vee \bigvee_{\mathsf{j}=1}^{5} \mathsf{f}^{\mathsf{o}_\mathsf{j}}.$$

Dans ce cas particulier, comme les quatre variables peuvent coder 16 informations (et non 9), il faut s'assurer de toujours manipuler des formules admissibles (dont les modèles ne sont que des modèles codant des événements). Tout sous ensemble de Σ sera ainsi codé par la conjonction de f^Σ (garantissant ainsi des modèles admissibles) et d'une disjonction de $\mathsf{f}^{\mathsf{e}_\mathsf{i}}$. Par exemple, l'ensemble $\{o_1, o_3, o_5\}$ sera représenté par

$$\mathsf{f}^{\{\mathsf{o}_1,\mathsf{o}_3,\mathsf{o}_5\}} \equiv (\mathsf{f}^{\mathsf{o}_1} \vee \mathsf{f}^{\mathsf{o}_3} \vee \mathsf{f}^{\mathsf{o}_5}) \wedge \mathsf{f}^\Sigma.$$

Le même procédé est utilisé pour représenter la formule des états $\mathsf{f}^{\mathsf{Q}_\mathsf{i}}$ associé à l'ensemble Q_i. La formule $\mathsf{f}^{\mathsf{Q}_\mathsf{i}}$ représente plus précisément l'ensemble des états sources de transitions. L'ensemble des états cibles est quant à lui représenté par la formule $\mathsf{f}'^{\mathsf{Q}_\mathsf{i}}$ qui est équivalente à $\mathsf{f}^{\mathsf{Q}_\mathsf{i}}$ au renommage des variables booléennes près (toute variable $b_1^{Q_i}$ de $\mathsf{f}^{\mathsf{Q}_\mathsf{i}}$ est renommée en $b_1'^{Q_i}$ dans $\mathsf{f}'^{\mathsf{Q}_\mathsf{i}}$). La formule $\mathsf{f}^{\mathsf{q}_{\mathsf{o}_\mathsf{i}}}$ représente l'état initial. Dans l'ensemble des transitions de $\mathcal{A}_i$, on distingue les transitions observables $q \xrightarrow{e} q' \in T_{o_i}$ où e est un événement observable, les transitions fautives $q \xrightarrow{e} q' \in T_{f_i}$ où e est un événement de faute et les transitions de synchronisations $q \xrightarrow{e} q' \in T_{s_i}$, où e est un événement de synchronisation. Une transition $t = q_1 \xrightarrow{e} q_2$ est codée par la conjonction :

$$\mathsf{f}^{\mathsf{t}} \equiv \mathsf{f}^{\mathsf{q}_1} \wedge \mathsf{f}^{\mathsf{e}} \wedge \mathsf{f}'^{\mathsf{q}_2}$$

où $\mathsf{f}^{\mathsf{q}_1}$ résulte du codage source de $\mathsf{f}^{\mathsf{Q}_\mathsf{i}}$ ($\mathsf{f}^{\mathsf{q}_1} \models \mathsf{f}^{\mathsf{Q}_\mathsf{i}}$) et $\mathsf{f}'^{\mathsf{q}_2}$ résulte du codage cible de $\mathsf{f}'^{\mathsf{Q}_\mathsf{i}}$ ($\mathsf{f}'^{\mathsf{q}_2} \models \mathsf{f}'^{\mathsf{Q}_\mathsf{i}}$). À titre d'illustration, les transitions de l'automate $\mathcal{A}_3$ de la figure 3.1 sont présentées dans la table 3.1 avec le codage suivant $z_1 \longrightarrow 0, z_2 \longrightarrow 1, z_3 \longrightarrow 2$ pour les états Q_3 de $\mathcal{A}_3$ et $f_1 \longrightarrow 0, f_2 \longrightarrow 1, c_1 \longrightarrow 2, c_2 \longrightarrow 3, o_1 \longrightarrow 4, o_2 \longrightarrow 5, o_3 \longrightarrow 6, o_4 \longrightarrow 7, o_5 \longrightarrow 8$ pour les événements de $\Sigma \supset \Sigma_3$. Il est à noter qu'il existe plusieurs choix de codages possibles. Par la suite, afin de simplifier les notations symboliques, les formules booléennes telles que $\mathsf{f}^{\mathsf{q}_1}, \mathsf{f}^{\mathsf{T}_{\mathsf{s}\mathsf{i}}}, \dots$ seront notées de façon ensembliste dans la même police de caractère, ainsi $\mathsf{f}^{\mathsf{q}_1}$ sera notée q_1 (représentation symbolique source de $q_1 \in Q_1$), $\mathsf{f}'^{\mathsf{q}_1}$ sera notée q_1' (représentation symbolique cible de q_1). Avec cette simplification, un composant symbolique sera ainsi noté :

$$sC_i = (\mathsf{Q}_\mathsf{i}, \mathsf{Q}_\mathsf{i}', \Sigma, \mathsf{q}_{\mathsf{o}_\mathsf{i}}, \mathsf{T}_{0_\mathsf{i}}, \mathsf{T}_{\mathsf{s}_\mathsf{i}}, \mathsf{T}_{\mathsf{f}_\mathsf{i}}).$$

Transitions	Formules
$z_1 \xrightarrow{c_1} z_2$	$\neg b_1^{Q_3} \wedge \neg b_2^{Q_3} \wedge f^{Q_3} \wedge \neg b_1^{\Sigma} \wedge \neg b_2^{\Sigma} \wedge b_3^{\Sigma} \wedge \neg b_4^{\Sigma} \wedge f^{\Sigma} \wedge \neg b_1'^{Q_3} \wedge b_2'^{Q_3} \wedge f'^{Q_3}$
$z_2 \xrightarrow{o_4} z_1$	$\neg b_1^{Q_3} \wedge b_2^{Q_3} \wedge f^{Q_3} \wedge \neg b_1^{\Sigma} \wedge b_2^{\Sigma} \wedge b_3^{\Sigma} \wedge b_4^{\Sigma} \wedge f^{\Sigma} \wedge \neg b_1'^{Q_3} \wedge \neg b_2'^{Q_3} \wedge f'^{Q_3}$
$z_1 \xrightarrow{c_2} z_3$	$\neg b_1^{Q_3} \wedge \neg b_2^{Q_3} \wedge f^{Q_3} \wedge \neg b_1^{\Sigma} \wedge \neg b_2^{\Sigma} \wedge b_3^{\Sigma} \wedge b_4^{\Sigma} \wedge f^{\Sigma} \wedge b_1'^{Q_3} \wedge \neg b_2'^{Q_3} \wedge f'^{Q_3}$
$z_3 \xrightarrow{o_5} z_1$	$b_1^{Q_3} \wedge \neg b_2^{Q_3} \wedge f^{Q_3} \wedge b_1^{\Sigma} \wedge \neg b_2^{\Sigma} \wedge \neg b_3^{\Sigma} \wedge \neg b_4^{\Sigma} \wedge f^{\Sigma} \wedge \neg b_1'^{Q_3} \wedge \neg b_2'^{Q_3} \wedge f'^{Q_3}$

TABLE 3.1 – Ensemble des transitions symboliques du composant C_3 (figure 3.1).

Il reste maintenant à définir la synchronisation de transitions d'automates de façon symbolique. Tout repose sur le fait que le codage des événements de chaque composant symbolique est global : les variables booléennes codant tous les événements sont les mêmes dans chaque composant symbolique. Soit deux automates sC_1 et sC_2, soit $T_i(e)_{i\in\{1,2\}}$ l'ensemble des transitions symboliques de sC_i définies comme suit :

$$T_i(e) \stackrel{\text{def}}{=} \begin{cases} (T_{0_i} \vee T_{s_i} \vee T_{f_i}) \wedge e & \text{si } e \in \Sigma_i, \\ \bigvee_{q_i^j \models Q_i} q_i^j \wedge q_i'^j & \text{si } e \notin \Sigma_i. \end{cases}$$

Intuitivement $T_i(e)$ regroupe toutes les transitions de sC_i étiquetées par e si e fait partie du composant sC_i et sinon elle contient l'ensemble des transitions fictives *sans événements* exprimant que sC_i ne change pas d'états dès lors que e se produit sur un autre composant. On obtient alors l'ensemble des transitions synchronisées par e de $\mathcal{A}_1 \| \mathcal{A}_2$ par la simple conjonction :

$$T_1(e) \wedge T_2(e).$$

En effet, si e est commun aux deux composants, la conjonction va naturellement synchroniser les deux ensembles de transitions. Si e n'appartient qu'à un seul composant, la conjonction exprime que le composant influencé par e change effectivement d'état alors que l'autre n'évoluera pas (transition fictive) peu importe son état courant. L'ensemble des transitions du produit $\mathcal{A}_1 \| \mathcal{A}_2$ est donc obtenu de la façon suivante :

$$T_1 \| T_2 \stackrel{\text{def}}{=} \bigvee_{e\in\Sigma} T_1(e) \wedge T_2(e).$$

Là encore, l'extension de cette opération à n composants ne pose aucune difficulté. L'ensemble des transitions résultant de la synchronisation des n composants d'un système est donné par :

$$\mathsf{T}_1 \| \ldots \| \mathsf{T}_n \stackrel{\text{def}}{=} \bigvee_{e \in \Sigma} \mathsf{T}_1(e) \wedge \cdots \wedge \mathsf{T}_n(e).$$

L'aptitude à manipuler efficacement des ensembles de transitions a permis de développer un *spectre d'algorithmes* (transfert de complexité spatio-temporelle) allant d'un algorithme symbolique partant du modèle initial, sans précompilation d'information, (algorithme basé composants) à un algorithme symbolique avec précompilation maximale (diagnostiqueur symbolique).

3.2.4 Diagnostic basé sur le modèle compositionnel

Le premier algorithme proposé [63] part de la modélisation symbolique compositionnelle (voir l'algorithme 3.1). Il ne fait aucune précompilation d'information, tout le traitement de diagnostic est fait lors de l'analyse de la séquence d'observation OBS. La version présentée dans [63],[64] et reprise ici est une version dite *en-ligne* ou encore *incrémentale*, à savoir que l'algorithme prend en entrée un état de croyance X issu d'un diagnostic précédent (sur les observations OBS′) et le met à jour avec l'apparition d'un nouvel événement observé o (OBS = OBS′.o). Le principe de l'algorithme est le suivant. Il se présente sous la forme d'une itération qui accumule à partir de X l'ensemble de transitions synchronisées et non observables issues du produit des composants. Une fois cette accumulation terminée, on cherche à calculer l'ensemble des transitions étiquetées par l'observation o qui émanent de cette accumulation, l'état cible de chacune de ces transitions faisant ainsi partie du résultat final X. L'algorithme met en jeu deux opérations importantes sur les BDDs à savoir, l'abstraction existentielle (fonction *absExistentielle*) et l'échange de variables (fonction *échange*). La fonction absExistentielle(X, Y) considère le BDD X et va éliminer de X la présence de toute variable qui n'est pas présente dans Y. La fonction échange(X, $\{\mathsf{b}_1 \ldots \mathsf{b}_l\}$) traduit X sur les variables booléennes $\{\mathsf{b}_1 \ldots \mathsf{b}_l\}$ (la formule X est ainsi recodée sur les variables booléennes $\{\mathsf{b}_1 \ldots \mathsf{b}_l\}$).

Le principe de l'itération (lignes 3-13) est d'accumuler dans X′ l'ensemble des états-fautes (ensemble de formules du type x ∧ E représentant un couple (x, E) où x est un état du système et E est un ensemble de fautes) issus d'un chemin de transitions non-observables venant d'un état de X, à chaque pas d'itération Y contient les états nouvellement déterminés. Pour cela, on tire d'abord à partir de X′ l'ensemble des transitions non observables pour chaque composant (lignes 4-7) qui sont ensuite synchronisées (ligne 8). En ligne 9, la fonction *propagationFautes* reprend Y et propage sur les états-fautes cibles de T_{nouv} les fautes contenues dans les états-fautes de Y. On récupère ensuite les états-fautes cibles (ligne 9) qui sont convertis en états-fautes sources (ligne 10) et ajoutés ainsi à X′ (ligne 11-12). La ligne 14 calcule l'ensemble des transitions observables dont l'étiquette est o qui peuvent être franchies à partir d'un état de X′, étiquette o qui, par la suite est supprimée de $\mathsf{T}_{\mathsf{obs}}$ et remplacée par les fautes de X′ (ligne 15). La

formule X_{obs} représente alors l'ensemble des états-fautes issus de l'observation de o à partir de X avec un codage cible qui est ensuite converti en codage source (ligne 17), pour obtenir ainsi le résultat escompté.

1 **Fonction** *Diagnostiquer*($\{sC_i\}_{i\in\{1,\dots,n\}}$, X, *o*)
Entrées : $sC_i = (Q_i, Q'_i, \Sigma, q_{o_i}, T_{0_i}, T_{s_i}, T_{f_i})$ composant symbolique
Entrées : $X = \bigvee(\bigwedge_{i=1}^{n} x^i \wedge E_f^i)$
2 $Y \leftarrow X$; $X' \leftarrow X$;
3 **tant que** $Y \not\equiv \bot$ **faire**
4 **pour tous les** $i \in \{1, \dots, n\}$ **faire**
5 $X_i \leftarrow$ absExistentielle(Y, Q_i);
6 $T^i_{nouv} \leftarrow (T_{si} \vee T_{fi}) \wedge X_i$;
7 **fin pour tous**
8 $T_{nouv} \leftarrow T^1_{nouv} \| \dots \| T^n_{nouv}$;
9 $X_{cible} \leftarrow$ propagationFautes(Y, T_{nouv});
10 $X_{source} \leftarrow$ échange($X_{cible}, \bigcup_{i=1}^{n} b^{Q_i}, \bigcup_{i=1}^{n} b^{Q'_i}$);
11 $Y \leftarrow X_{source} \wedge \neg X'$;
12 $X' \leftarrow X' \vee Y$;
13 **fin tq**
14 $T_{obs} \leftarrow$ absExistentielle($X', \bigvee_{i=1}^{n} Q_i$) $\wedge$ o $\wedge \bigvee_{i=1}^{n} T_{oi}$;
15 $T_{obs} \leftarrow$ absExistentielle($T_{obs}, \bigvee_{i=1}^{n} Q_i \vee \bigvee_{i=1}^{n} Q'_i$) $\wedge X'$;
16 $X_{obs} \leftarrow$ absExistentielle($X_{obs}, \Sigma \vee \bigvee_{i=1}^{n} Q'_i$);
17 $X \leftarrow$ échange($X', \bigcup_{i=1}^{n} b^{Q_i}, \bigcup_{i=1}^{n} b^{Q'_i}$);
Résultat : X

Algorithme 3.1 : Diagnostic symbolique basé sur le modèle compositionnel.

3.2.5 Diagnostic basé sur le modèle global symbolique

Dans l'algorithme 3.1, la synchronisation des composants se fait en-ligne. Cette opération étant coûteuse, on peut décider dans la mesure du possible de la précompiler en calculant au préalable à toute tâche de diagnostic la représentation symbolique *sS* du système.

$$sS \stackrel{\text{def}}{=} (\bigvee_{i=1}^{n} Q_i, \bigvee_{i=1}^{n} Q'_i, \bigwedge_{i=1}^{n} q_{0i}, \Sigma, T_o, T_s, T_f)$$

où $T_o = T_{o1} \| \dots \| T_{on}$, $T_s = T_{s1} \| \dots \| T_{sn}$ et $T_f = T_{f1} \| \dots \| T_{fn}$.

L'algorithme 3.1 est donc adapté en l'algorithme qui utilise quant à lui le modèle global *sS* (voir figure 3.4).

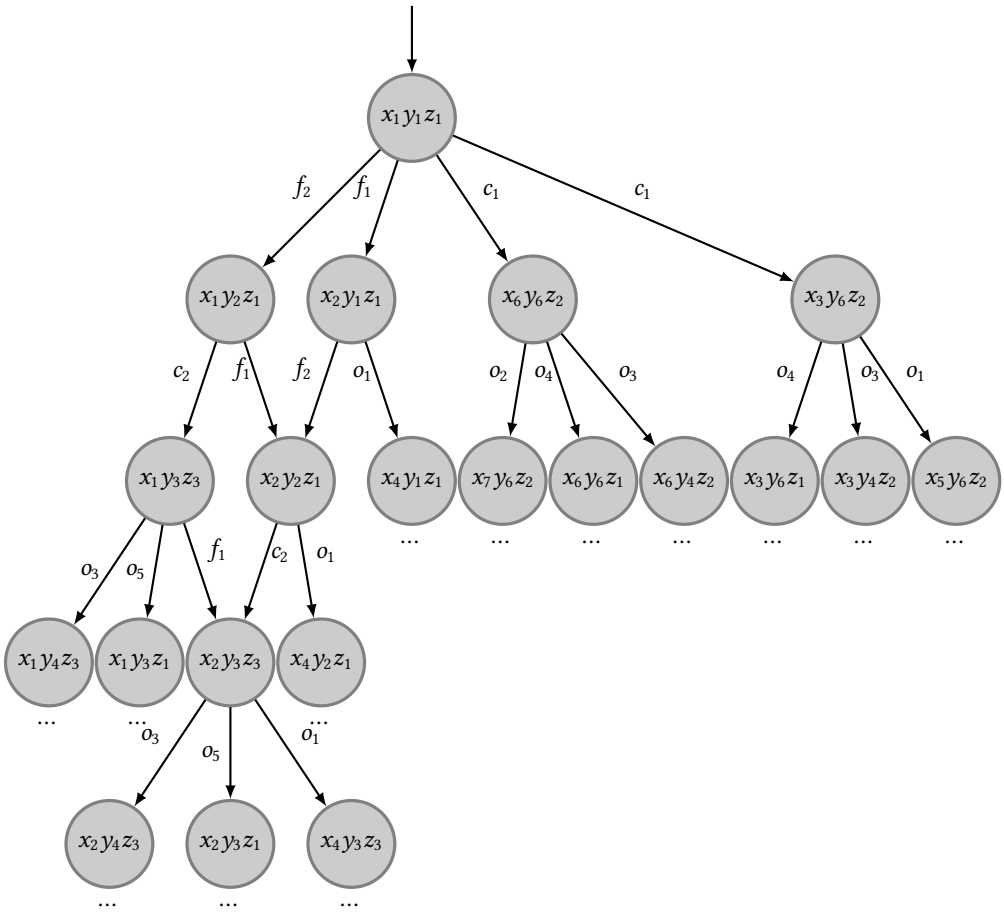


FIGURE 3.4 – Extrait du modèle global $\mathcal{A}_1 \| \mathcal{A}_2 \| \mathcal{A}_3$ (voir figure 3.1) de l'état initial à la première transition observable.

Fonction *Diagnostiquer*(sS, X, o)
Entrées : $sS = (\mathsf{Q}, \mathsf{Q}', \Sigma, \mathsf{q_o}, \mathsf{T_0}, \mathsf{T_s}, \mathsf{T_f})$ modèle global symbolique
Entrées : $\mathsf{X} = \bigvee(\mathsf{x} \wedge \mathsf{E_f})$
$\mathsf{Y} \leftarrow \mathsf{X}$; $\mathsf{X}' \leftarrow \mathsf{X}$;
tant que $\mathsf{Y} \not\equiv \bot$ **faire**
  $\mathsf{T_{nouv}} \leftarrow \mathsf{Y} \wedge (\mathsf{T_s} \vee \mathsf{T_f})$;
  $\mathsf{X_{cible}} \leftarrow$ propagationFautes($\mathsf{Y}, \mathsf{T_{nouv}}$);
  $\mathsf{X_{source}} \leftarrow$ échange($\mathsf{X_{cible}}$,$\mathsf{b^Q}$,$\mathsf{b^{Q'}}$);
  $\mathsf{Y} \leftarrow \mathsf{X_{source}} \wedge \neg\mathsf{X}'$;
  $\mathsf{X}' \leftarrow \mathsf{X}' \vee \mathsf{Y}$;
fin tq
$\mathsf{T_{obs}} \leftarrow$ absExistentielle(X', Q) $\wedge$ o $\wedge$ $\mathsf{T_o}$;
$\mathsf{T_{obs}} \leftarrow$ absExistentielle($\mathsf{T_{obs}}, \mathsf{Q} \vee \mathsf{Q}'$) $\wedge$ X';
$\mathsf{X_{obs}} \leftarrow$ absExistentielle($\mathsf{X_{obs}}, \Sigma \vee \mathsf{Q}'$);
$\mathsf{X} \leftarrow$ échange($\mathsf{X}', \mathsf{b^Q}, \mathsf{b^{Q'}}$);
Résultat : X

Algorithme 3.2 : Diagnostic symbolique basé sur le modèle global.

3.2.6 Diagnostic basé sur une machine symbolique abstraite

L'algorithme 3.1 propose une première phase de précompilation en précalculant le modèle global du système afin d'éviter, en-ligne, de synchroniser les transitions de composants. La deuxième phase de précompilation porte sur une autre opération coûteuse : *propagationFautes*. Étant donné un état de croyance X, l'objectif de *propagationFautes* est le suivant. Cette fonction analyse *tous* les chemins non observables issus de X et menant à la nouvelle observation o et sur chacun de ces chemins, elle doit en extraire l'ensemble des événements fautifs, les fusionner avec les fautes déjà associées à l'état source du chemin dans X afin de propager le tout et de l'associer à l'état final du chemin qui fait partie du nouvel état de croyance. L'objectif de ce troisième algorithme est de précompiler en partie l'information résultant de *propagationFautes* dans une machine à états abstraite *sSA* (voir figure 3.5). À partir d'un état x du modèle *sS*, (état initial ou état cible d'une observation), on précompile une abstraction de tous les chemins non observables issus de x contenant exactement l'ensemble de fautes F. Sur la figure 3.5, les ensembles de fautes possibles sont $\emptyset$, $\{f_1\}$, $\{f_2\}$ et $\{f_1, f_2\}$. Une fois cette abstraction calculée, on créé un état fictif à partir duquel on reproduit l'ensemble des transitions observables du modèle global pouvant se produire à partir de x en franchissant tout d'abord un chemin de transitions non observables contenant exactement les fautes F puis la transition observable en question. L'abstraction réduit ainsi, en-ligne, le calcul des chemins non observables issus de x contenant exactement un ensemble de fautes F au déclenchement d'une seule transition. Cette machine abstraite est for-

mellement un sextuplet de formules :

$$sSA \stackrel{\text{def}}{=} (\mathsf{Q}^{\mathsf{A}}, \mathsf{Q}'^{\mathsf{A}}, \Sigma^{\mathsf{A}}, \mathsf{q}_0^{\mathsf{A}}, \mathsf{T}_{\mathsf{o}}^{\mathsf{A}}, \mathsf{T}_{\mathsf{f}}^{\mathsf{A}})$$

où $\mathsf{Q}^{\mathsf{A}}, \mathsf{Q}'^{\mathsf{A}}$ codent l'ensemble des états de la machine abstraite (versions source et cible) : cet ensemble contient les états fictifs nécessaires, les états du modèle global cibles d'une transition observable et l'état initial du système (q_0^A). Σ^{A} est l'alphabet de la machine, il code les événements observables du modèle global ainsi que l'ensemble des ensembles de fautes (Σ^{A} code l'ensemble $\bigcup_{i=1}^{n} \Sigma_{oi} \cup 2^{\bigcup_{i=1}^{n} \Sigma_{fi}}$). Finalement $\mathsf{T}_{\mathsf{o}}^{\mathsf{A}}$ et $\mathsf{T}_{\mathsf{f}}^{\mathsf{A}}$ codent respectivement les transitions observables et fautives de la machine abstraite.

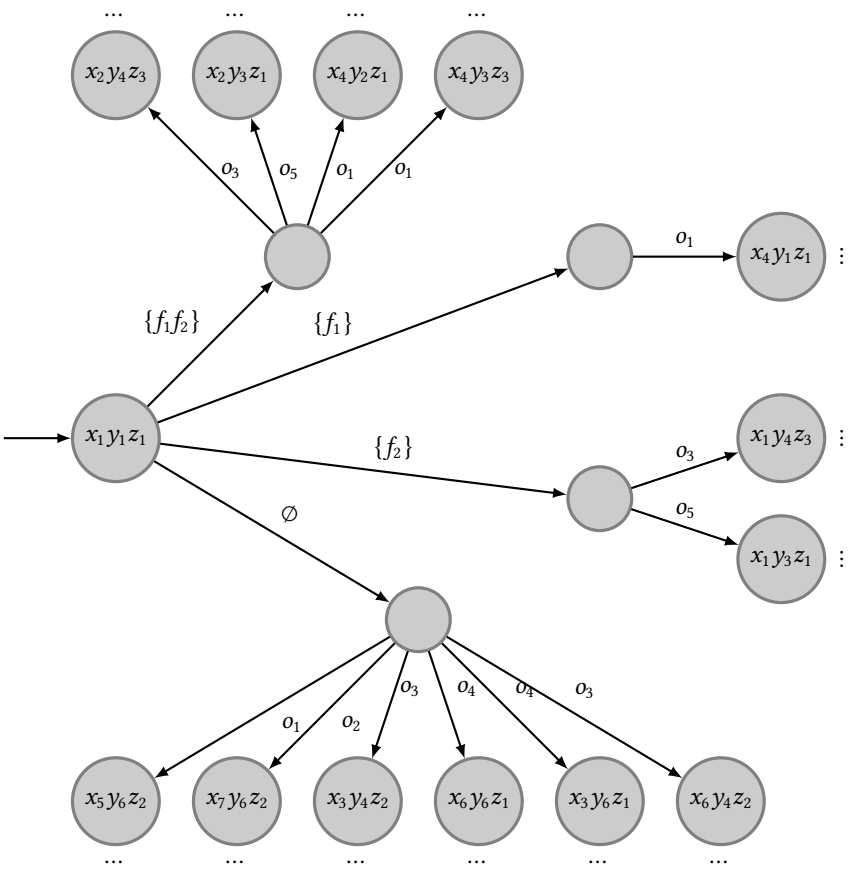


FIGURE 3.5 – Extrait de la machine abstraite du modèle global $\mathcal{A}_1 \| \mathcal{A}_2 \| \mathcal{A}_3$ (voir figure 3.4) de l'état initial à la première transition observable.

L'algorithme 3.3 présente le calcul du diagnostic en s'appuyant sur la machine abstraite. L'itération présente dans l'algorithme 3.2 a disparu dans cette version car le calcul des chemins non-observables issus de X est désormais mis en œuvre par la seule ligne 2 où les transitions fautives issues de X sont toutes explorées en même temps. Les

1 **Fonction** *Diagnostiquer*(sSA, X, o)
Entrées : $sSA = (Q^A, Q'^A, \Sigma^A, q_o^A, T_o^A, T_f^A)$ machine abstraite symbolique
Entrées : $X = \bigvee(x \wedge E_f)$
2 $T_{nouv} \leftarrow X \wedge (T_f^A)$;
3 $X_{cible} \leftarrow$ propagationFautesAbstrait(Y, T_{nouv});
4 $X' \leftarrow$ échange(X_{cible},b^{Q^A},$b^{Q'^A}$);
5 $T_{obs} \leftarrow$ absExistentielle(X', Q^A) $\wedge$ o $\wedge$ T_o^A;
6 $T_{obs} \leftarrow$ absExistentielle($T_{obs}, Q^A \vee Q'^A$) $\wedge$ X';
7 $X_{obs} \leftarrow$ absExistentielle($X_{obs}, \Sigma \vee Q'^A$);
8 $X \leftarrow$ échange($X', b^{Q^A}, b^{Q'^A}$);
Résultat : X

Algorithme 3.3 : Diagnostic symbolique basé sur la machine abstraite.

états de X' (ligne 4) rassemblent alors tous les états fictifs (additionnés des fautes propagées par propagationFautesAbstrait) issus des franchissements possibles de transitions fautives issues de X. Le résultat final s'obtient alors de la même manière que dans l'algorithme 3.2 (lignes 5-8).

3.2.7 Diagnostic basé sur un diagnostiqueur symbolique

La machine abstraite décrite ci-dessus est une machine qui à partir d'un état x représente par différents chemins de transitions, l'ensemble des ensembles fautifs qui peuvent survenir si une observation *o* doit être expliquée à partir cet état. En général, l'estimation de l'état x par le diagnostic est incertaine (plusieurs états candidats) ce qui conduit l'algorithme 3.3 à parcourir un ensemble de transitions sur la machine abstraite qui dépend du nombre d'états candidats à un instant donné pour traiter chaque nouvelle observation. De plus, il est toujours nécessaire de propager les fautes passées pour avoir un diagnostic correct. Ce sont ces deux traitements que le diagnostiqueur symbolique précompile. Le diagnostiqueur symbolique est une représentation symbolique du diagnostiqueur [Sampath et al., 1996] (voir la figure 3.6 pour en voir un extrait).

Le diagnostiqueur est une machine à états déterministe dont les transitions sont étiquetées par les observables. Cette machine est la précompilation optimale du problème de diagnostic. On en propose ici une version symbolique qui a pour objectif de compacter en taille cette machine en la faisant bénéficier du codage efficace des BDDs.

Le diagnostiqueur symbolique se présente comme un quintuplet

$$s\Delta = (Q^{s\Delta}, Q'^{s\Delta}, \Sigma^{s\Delta}, q_o^{s\Delta}, T_o^{s\Delta})$$

où $Q^{s\Delta}$ et $Q'^{s\Delta}$ codent respectivement les versions sources et cibles des états du diagnostiqueur. Ici le codage est plus complexe car tout état du diagnostiqueur est associé

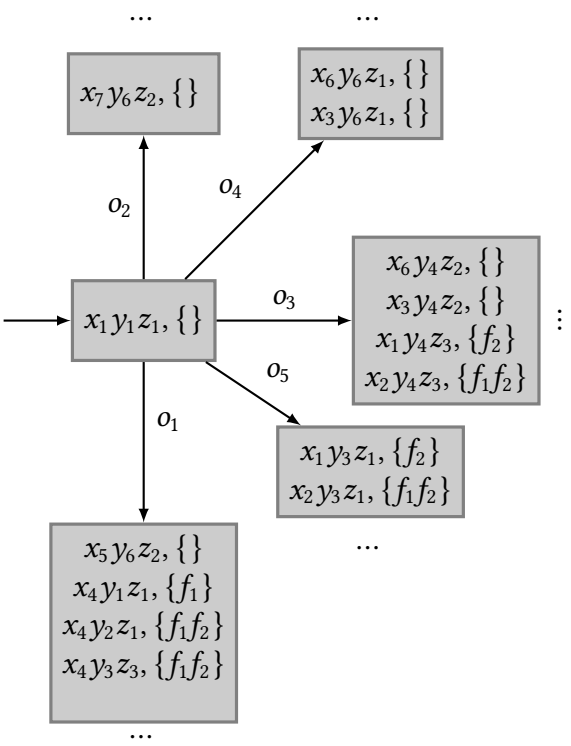


FIGURE 3.6 – Extrait du diagnostiqueur du modèle global $\mathcal{A}_1 \| \mathcal{A}_2 \| \mathcal{A}_3$ (voir figure 3.4) de l'état initial à la première transition observable.

à un sous-ensemble de $2^{(Q\times 2^{\Sigma_f})}$. La formule $\Sigma^{s\Delta}$ code les événements observables Σ_o uniquement et $T_o^{s\Delta}$ codent l'ensemble des transitions observables du diagnostiqueur. L'état initial $q_o^{s\Delta}$ code l'état singleton $\{(q_o, \emptyset)\}$ où q_0 est l'état initial du modèle global.

L'algorithme 3.4 décrit l'utilisation du diagnostiqueur symbolique pour résoudre le problème de diagnostic. Il est réduit à deux lignes. La ligne 2 correspond au tirage effectif de la transition étiquetée par o issue de l'état X et la ligne 3 est la reproduction sur le codage source de l'état cible de cette transition. On peut noter donc que le calcul effectif sur le diagnostiqueur symbolique est plus coûteux que le simple tirage d'une transition sur un diagnostiqueur énumératif (diagnostiqueur codé de façon énumérative sur une machine). Mais comme nous le verrons dans la partie expérimentale, ce coût est largement compensé par le gain mémoire de la représentation symbolique.

1 **Fonction** *Diagnostiquer*($s\Delta$, X, o)
  Entrées : $s\Delta = (Q^{s\Delta}, Q'^{s\Delta}, \Sigma^{s\Delta}, q_o^{s\Delta}, T_o^{s\Delta})$ diagnostiqueur symbolique
  Entrées : $X = \bigvee(x \wedge E_f) \in Q^{s\Delta}$
2 $X' \leftarrow$ absExistentielle($X \wedge o \wedge T_o^{s\Delta}, Q'^{s\Delta}$);
3 $X \leftarrow$ échange($X', b^{Q^{s\Delta}}, b^{Q'^{s\Delta}}$);
  Résultat : X

Algorithme 3.4 : Diagnostic symbolique basé sur le diagnostiqueur symbolique.

3.2.8 Quelques résultats expérimentaux

Dans cette section, nous présentons quelques résultats expérimentaux qui illustrent le comportement des algorithmes précédemment décrits.

3.2.8.1 Comparatifs diagnostiqueurs symbolique et énumératif

Dans un premier temps, nous illustrons la comparaison entre le diagnostiqueur symbolique, décrit en section 3.2.7 et sa version énumérative [63]. La version énumérative du diagnostiqueur est une structure de données où tous les états et toutes les transitions (voir figure 3.6) sont explicitement représentés en mémoire. Afin d'illustrer les complexités spatiales et temporelles sous-jacentes, un ensemble de modèles $M_1 \ldots M_4$ ont été établis avec une taille croissante en nombre d'états et de transitions (voir table 3.2).

La figure 3.7 présente les temps de calcul respectifs pour la synthèse hors-ligne (précompilation) des diagnostiqueurs symboliques et énumératifs pour les 4 modèles générés. L'échelle logarithmique illustre la nature de la complexité de cette synthèse à savoir exponentielle en la taille des modèles. Elle illustre également l'effet du calcul symbolique sur le temps effectif de synthèse d'un diagnostiqueur. Ceci s'explique par

	M_1	M_2	M_3	M_4
États	353	921	2500	4355
Transitions	2183	5774	16530	31024

TABLE 3.2 – Taille des modèles utilisées.

le phénomène de factorisation des calculs (traitement de plusieurs transitions en une seule fois).

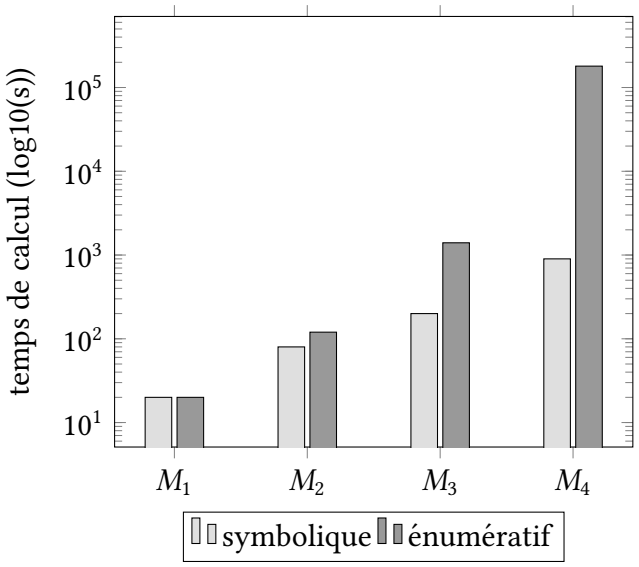


FIGURE 3.7 – Comparaison du temps de calcul pour la synthèse d'un diagnostiqueur en fonction de la taille des modèles.

Cet effet factorisant est également visible sur la taille des structures ainsi synthétisées comme le présente la figure 3.8. Cette figure illustre également la complexité spatiale exponentielle d'un algorithme de type diagnostiqueur qui est fonction de la taille des modèles.

Le gain temporel de la synthèse augmente avec la taille du modèle, il s'explique, en plus de l'effet factorisant du calcul symbolique, par l'utilisation de cache mémoire dans l'utilisation des BDDs : ce cache permet, en pratique, de retrouver bien plus efficacement le fait qu'un ensemble d'états donnés du diagnostiqueur a déjà été construit/revisité, que dans la version énumérative. Le gain spatial, quant à lui, s'explique par l'effet factorisant de la représentation du diagnostic contenu dans chaque état du diagnostiqueur. Comme il l'a été précédemment énoncé, le diagnostiqueur classique est la structure de données la plus compilée et qui permet de résoudre le problème

de diagnostic en n opérations en temps constant (n tirages de transitions pour une séquence d'observations de n événements). Dès lors que la représentation symbolique de ce même diagnostiqueur est drastiquement plus petite, cela signifie que cette structure n'est pas la plus compilée. En effet, comme nous le verrons dans la section suivante (figure 3.10), l'utilisation du diagnostiqueur symbolique n'est pas constituée de n opérations en temps constant, car le diagnostic de l'état cible obtenu après tirage des n événements doit être énuméré à partir de la formule logique qui le représente, opération dont la complexité temporelle est au pire exponentielle en le nombre de variables codant le diagnostic de cet état.

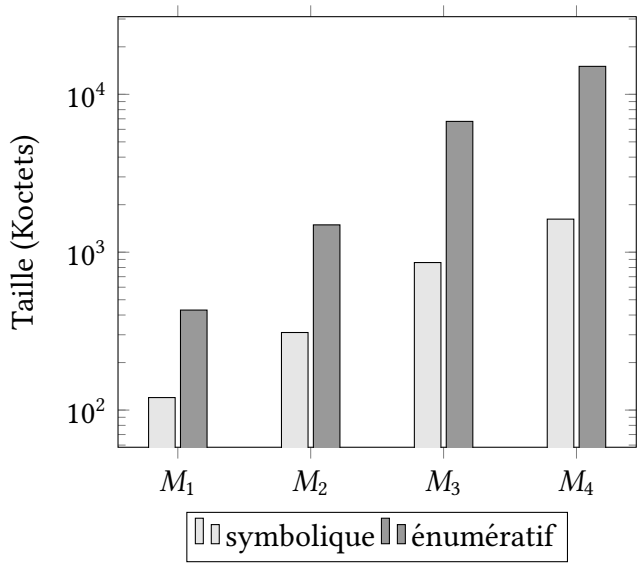


FIGURE 3.8 – Comparaison des tailles de diagnostiqueurs en fonction de la taille des modèles.

3.2.8.2 Comparatifs des algorithmes du spectre proposé

Le premier comparatif est présenté sur la figure 3.9. À partir d'un composant communicant, on construit un jeu de systèmes de type $n \times m$ composants (une grille de n par m composants du même type) avec la garantie que la taille effective du modèle global croît exponentiellement en le nombre de composants constitutifs du système. La figure 3.9 compare, pour un jeu de 5 systèmes créés de cette façon, le temps moyen de calcul des algorithmes compositionnels énumératifs et symboliques sur 10 scénarios observables de 1000 observations chacun. Avec une approche de type compositionnelle, rien n'est précompilé et donc la complexité temporelle de l'approche (symbolique ou non) est exponentielle en la taille des modèles. On voit néanmoins ici l'intérêt de l'approche symbolique qui offre de meilleurs temps de calcul sachant que pour l'approche

énumérative les résultats pour les systèmes 1 × 5 et 2 × 3 ne sont tout simplement pas disponibles car ils ont demandé beaucoup trop de temps de calcul. [1] Cette figure illustre deux points importants :

1. le premier est l'apport du symbolique (augmentation de la mise à l'échelle par rapport au calcul énumératif) ;
2. le second est le besoin de méthodes précompilées afin d'éviter le problème de la complexité temporelle exponentielle de l'approche compositionnelle.

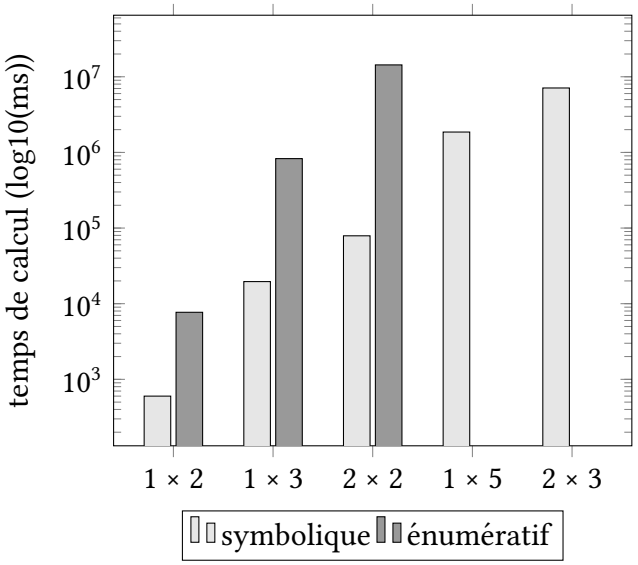


FIGURE 3.9 – Comparaison du temps de calcul pour le diagnostic sur un modèle compositionnel.

La figure 3.10 illustre enfin les temps de calcul du diagnostic pour un modèle donné qui est constitué de 3 composants communicants et dont les propriétés du modèle global sont présentées en table 3.4. Pour obtenir cette figure, 100 scénarios de 10000 observations chacun ont été générés et toutes les méthodes ont été utilisées sur ces 100 scénarios afin de déterminer leur temps moyen de calcul respectif. Cette figure illustre plusieurs choses. La première concerne les approches énumératives. L'échelle logarithmique montre le transfert de complexité exponentielle du temporel vers le spatial entre la méthode compositionnelle énumérative et la méthode de diagnostiqueur classique. Entre la méthode compositionnelle et la méthode globale, on transfère le

1. L'approche symbolique ne casse bien évidemment pas la complexité du problème du diagnostic centralisé mais l'augmentation des performances en pratique peut s'avérer utile notamment dès que l'on cherche à résoudre le problème de façon décentralisée ou distribuée (voir notamment la section 3.4) sur le sujet.

calcul des synchronisations, entre la méthode globale et abstraite, on y transfère la recherche de chemins non-observables, et enfin entre la méthode abstraite et la méthode du diagnostiqueur, on y transfère la propagation des fautes et le comportement non-déterministe. Le comportement de ce transfert sur les méthodes symboliques n'est pas identique. La table 3.3 présente des mesures quantitatives de ce transfert, on y établit pour chaque algorithme symbolique, le temps de calcul consacré au tirage effectif des transitions observables et le temps consacré à des calculs qui pourraient être précompilés. Il semble par exemple que la méthode compositionnelle et la méthode globale aient des performances temporelles identiques ce qui tend à montrer que la méthode globale n'est pas intéressante. Ce phénomène s'explique très bien, la méthode compositionnelle bénéficie d'un cache, dès lors qu'une synchronisation a été effectuée elle est conservée si bien que la méthode compositionnelle a tendance à se comporter à la limite comme une méthode globale où toutes les synchronisations ont déjà été calculées (dans la limite de la taille du cache physique disponible). La comparaison entre chaque approche symbolique et énumérative s'explique également très bien. Pour les 3 premières méthodes (compositionnelle, globale et abstraite), elle bénéficie des avantages du calcul symbolique à savoir la représentation implicite d'un ensemble d'états et/ou de transitions et l'utilisation d'un cache mémoire intrinsèque à l'utilisation des BDDs. Pour la dernière méthode, c'est l'inverse, en effet la représentation implicite devient un inconvénient car le résultat final du diagnostic doit être énuméré à partir de la formule résultat d'où une perte de performance. [2]

	Compos.	Global	Abstrait	Diag.
Précompilation	94.17%	94, 9%	81, 61%	0%
Tirage effectif de transitions	5, 83%	5, 10%	18, 39%	100%

TABLE 3.3 – Comparaisons des pourcentages de temps de précompilation pour tous les algorithmes symboliques du spectre.

Pour terminer cette étude comparative, la table 3.4 présente les ressources mémoires nécessaires pour stocker les machines nécessaires à l'utilisation de chaque algorithme (le cache n'est pas mesuré ici). Dès lors que l'on utilise des machines fortement précompilées (méthodes abstraites et diagnostiqueurs), on peut constater un gain mémoire intéressant et plus particulièrement pour le diagnostiqueur symbolique.

2. Cette perte de performance ne se produit que si l'on veut énumérer, sur une console par exemple, l'ensemble des candidats comme le diagnostiqueur énumératif le ferait, mais on peut très bien imaginer qu'un système d'aide à la décision n'a besoin que de la représentation implicite et cette énumération devient alors inutile.

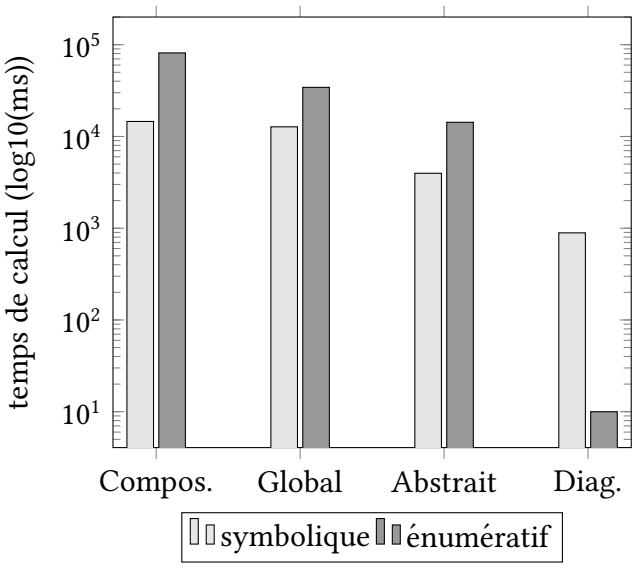


FIGURE 3.10 – Comparaison des temps de calcul du diagnostic de tous les algorithmes du spectre sur un même modèle.

	Compos.	Global	Abstrait	Diag.
États	moy : 17,7	1063	965	18474
Transitions	moy : 34	2912	48958	120698
Mémoire Mo (symbolique)	0.01	0.2	0.6	7.5
Mémoire Mo (énumératif)	0.01	0.2	2.7	123.9

TABLE 3.4 – Comparaison de la taille mémoire occupée par les structures de données.

3.2.8.3 Sélection de fenêtres temporelles

Les travaux sur le diagnostic symbolique ont été poursuivis dans le cadre des travaux de thèse de XingYu Su [Su and Grastien, 2013],[Su and Grastien, 2014], [67], [Su, 2016]. [3] La trame fondamentale des travaux de XingYu repose sur l'hypothèse que dans les systèmes complexes qui ont une *longue durée de vie*, il n'est pas nécessairement possible de suivre le flot entier d'observations et de mettre à jour un diagnostic avec une approche classique. L'idée ici a consisté à travailler sur des fenêtres temporelles en *oubliant totalement ou partiellement le reste*. Autrement dit est-il possible de diagnostiquer l'occurrence d'événements fautifs en n'observant que des fenêtres temporelles d'observations *en totale indépendance des autres*. XingYu a proposé dans ce cadre une série d'algorithmes IWA (*Independant Window Algorithms*) [Su and Grastien, 2013] où il montre par l'expérience qu'il y a de sérieuses pertes de précision [Su and Grastien, 2014]. C'est à partir de ce constat, qu'un compromis a été proposé dans [67]. Le principe a été de définir un *état de croyance abstrait* entre deux fenêtres temporelles et de l'utiliser comme une entrée supplémentaire d'un algorithme de diagnostic de type IWA (les algorithmes utilisant ces entrées sont appelés TWA (*Time Windows Algorithms*) dans [67] et [Su, 2016]). La mise en place d'un état de croyance abstrait a pour intérêt de conserver un minimum d'information entre les fenêtres sachant que l'information la moins abstraite (donc la plus raffinée) a une complexité spatiale exponentielle en le nombre de composants du système supervisé. Tous ces algorithmes ont été mis en œuvre par une approche symbolique (approche composant similaire à celle présentée dans la section 3.2.4) et comparés expérimentalement dans la thèse de XingYu Su [Su, 2016]. En particulier, on voit l'intérêt des algorithmes de type TWA par rapport à ceux de type IWA en terme de précisions mais cela se fait au prix d'une augmentation de la complexité temporelle des algorithmes.

3.2.9 Résumé

Les travaux de thèse d'Anika Schumann ont essentiellement porté sur le développement d'un ensemble d'algorithmes pour la résolution du problème de diagnostic centralisé en utilisant des techniques de calcul symbolique (BDDs). Ces travaux ont démontré l'intérêt de l'approche symbolique qui offre de meilleurs performances en termes de ressources mémoires et de temps de calcul. En outre, plusieurs algorithmes se situant à différents points du spectre de la complexité spatio-temporelle du problème initial ont été proposés afin de pouvoir sélectionner parmi eux le plus approprié pour

3. J'ai été membre officiel du comité de supervision de XingYu Su à l'université nationale d'Australie en tant que *Associate Supervisor*. À ce titre, j'ai guidé partiellement les travaux de XingYu et validé son travail final avec son encadrant principal (Alban Grastien) et son autre encadrant associé (Patrick Haslum,*Associate Supervisor*). Il est d'usage qu'un étudiant de l'ANU ne publie pas toujours avec ses encadrants associés, en particulier quand cet encadrement se fait à distance.

un problème donné. Un point intéressant notamment porte sur les performances de la méthode diagnostiqueur symbolique qui, de par ses temps de synthèse, sa consommation mémoire et ses performances en-ligne, permet de considérer définitivement des systèmes plus gros sur une architecture de diagnostic centralisée. Enfin, ces algorithmes servent également de base pour des approches de type décentralisé/distribué proposées dans [66]. Ils ont également permis de définir une famille d'algorithmes s'appuyant sur la présélection de fenêtres temporelles et la définition d'états abstraits de croyance développés dans les travaux de thèse de XingYu Su [67].

3.3 Diagnosticabilité des systèmes à événements discrets

La section précédente décrit les différentes contributions au problème du diagnostic de SED en proposant un spectre d'algorithmes symboliques qui permet de trouver le meilleur compromis entre la complexité spatiale et la complexité temporelle du problème de diagnostic pour un problème donné. Ces algorithmes tirent profit de toutes les observations du système, néanmoins ces observations peuvent s'avérer au final insuffisantes et les algorithmes peuvent éventuellement toujours retourner un ensemble de plusieurs candidats, donc un diagnostic ambigu. Les propriétés qui affirment qu'une méthode de diagnostic pourra conclure avec certitude avec une séquence d'observations *finie* sont des propriétés dites de *diagnosticabilité*.

3.3.1 Contextes des travaux

Les contextes de mes travaux sur l'analyse de la diagnosticabilité sont multiples. Ces analyses sont essentielles au succès d'une méthode de diagnostic car elles offrent au final un moyen d'améliorer les performances du diagnostic. L'analyse de la diagnosticabilité a démarré par une démarche personnelle [35] et [34] (section 3.3.3) où je développe un algorithme décentralisé de vérification de la diagnosticabilité afin de profiter de la structure compositionnelle des systèmes. Ces travaux ont ensuite été poursuivis par une contribution de thèse d'Anika Schumann dont l'objet était d'améliorer le passage à l'échelle des algorithmes développés précédemment (section 3.3.4). Afin de profiter de ces analyses pour améliorer la diagnosticabilité intrinsèque du système, un algorithme décentralisé de placements de capteurs a également été proposé dans le cadre de la thèse de Pauline Ribot (section 3.3.6.2). Les algorithmes précédemment cités résolvent le problème initial de la recherche de diagnosticabilité d'occurrence d'événements fautifs simples. Plus récemment, ces études de diagnosticabilité ont été reprises dans le cadre de la thèse de Houssam-Eddine Gougam en considérant des motifs d'intérêt plus complexes que la simple occurrence d'un événement de faute (section 3.3.5).

3.3.2 Définition du problème

La notion de diagnosticabilité des systèmes à événements discrets a été introduite dans [Lin, 1994] puis étendue dans [Sampath et al., 1995]. Cette première notion définit la diagnosticabilité de fautes simples (occurrence d'événements). Dans [Jéron et al., 2006], cette notion de diagnosticabilité a été étendue aux motifs d'intérêt (définition 3.9). C'est également à partir de cette définition sur les motifs que les travaux de Houssam-Eddine Gougam [25], [24], [26], [22] ont été développés.

Définition 3.21 (Diagnosticabilité d'un motif [25]) *Un motif μ est diagnosticable dans un système de modèle* DS *= $(\mathcal{S}, \Sigma_o, obs)$ si :*

$$\exists n \in \mathbb{N}, \forall \tau_1 \text{ une séquence } \mu\text{-concordante de } \mathcal{S}, \forall \tau_2 \in \mathcal{S}/\tau_1 :$$
$$|obs(\tau_2)| \geq n \implies (\forall \tau_3 \in \mathcal{S}, obs(\tau_3) = obs(\tau_1.\tau_2) \implies \tau_3 \text{ est } \mu\text{-concordante.})$$

On peut décrire intuitivement cette notion de la façon suivante. Supposons que le système génère une trajectoire τ_1 dans laquelle le motif μ a eu lieu. La μ-diagnosticabilité du système garantit qu'il suffit d'attendre la génération d'une continuation τ_2 de τ_1 produisant n observations pour avoir la *certitude* que le motif μ a eu lieu (toute trajectoire du système τ_3 générant la séquence observable de $\tau_1.\tau_2$ contient également le motif μ). Cette description intuitive nous montre également l'intérêt de cette propriété : pour un motif donné μ il existe un diagnostiqueur sur le système DS qui peut toujours conclure avec *certitude* sur l'occurrence du motif μ en attendant un nombre *fini* d'observations.

La question posée par la diagnosticabilité est la suivante : pour un motif μ, le système DS est-il diagnosticable ? La plupart des travaux sur cette question (sinon la totalité, les exceptions étant [Jiang and Kumar, 2004], [Jéron et al., 2006] et [25]) ne se sont intéressés qu'à la diagnosticabilité de fautes simples f (un motif réduit uniquement à l'occurrence d'un seul événement). Deux hypothèses sur le comportement du système sont faites pour résoudre cette question :

1. Le système est vivant : le système ne contient pas d'états de blocage, il génère des événements indéfiniment. Le cas où le système a des états de blocage est un cas dégénéré qui ne porte pas toute la complexité du problème général (des travaux existent sur le sujet notamment [Moreira et al., 2011]) ;
2. Le système ne génère pas de séquences non-bornées d'événements non-observables (typiquement représentées par des cycles d'événements non-observables dans le modèle global du système). Sans cette hypothèse, on n'a aucune garantie de la génération d'événements observables en un temps fini, ce qui conduit intrinsèquement à la non-diagnosticabilité du système.

La première solution proposée par [Sampath et al., 1995] consiste à analyser le diagnostiqueur (voir figure 3.6). En effet, cette machine précompilant tout le problème

contient la réponse à cette question. Pour qu'une faute f soit diagnosticable, il faut garantir qu'il n'existe pas dans le système DS deux trajectoires infinies τ_f et $\tau_{\neg f}$ telles que $f \in \tau_f$ et $f \notin \tau_{\neg f}$ ayant la même projection observable $obs(\tau_f) = obs(\tau_{\neg f})$. Cela conduit dans le diagnostiqueur à rechercher une séquence d'observations menant à un cycle d'états de croyance dans chacun desquels l'occurrence de la faute f est ambiguë (un diagnostic candidat affirmant que f a eu lieu et un autre affirmant le contraire). Cette première méthode a néanmoins deux inconvénients. Le premier est la complexité du calcul effectif du diagnostiqueur (voir section 3.2.2.2), le deuxième est que, par construction du diagnostiqueur, l'analyse ne se limite pas qu'à la simple recherche d'un cycle ambigu, encore faut-il que ce cycle retrace bien deux trajectoires du système, une analyse complémentaire est nécessaire pour confirmer la validité du cycle. Une deuxième approche, plus directe, est proposée dans [Jiang et al., 2001] et [Yoo and Lafortune, 2002], elle s'appuie sur le principe du produit jumelé du modèle global. Intuitivement, le modèle global est dupliqué et un produit de synchronisation est appliqué sur les deux clones (synchronisation sur les événements observables). Ce produit a pour effet de confronter chaque trajectoire du système à toute autre de son clone qui a la même projection observable. Une faute f est alors diagnosticable si, dans le produit jumelé, il n'existe pas de cycle impliquant une *paire critique* [Cimatti et al., 2003] à savoir une trajectoire contenant f et une trajectoire ne contenant pas f. L'avantage de cette méthode est qu'elle est moins complexe que la méthode du diagnostiqueur (polynomial en le nombre d'états du modèle global par opposition à la complexité exponentielle de l'approche diagnostiqueur) néanmoins, elle est exponentielle en le nombre de composants.

3.3.3 Analyse décentralisée

La première étude que j'ai réalisée sur l'analyse de diagnosticabilité part du constat que cette analyse est toujours vue de façon globale et centralisée [34] et donc on a un problème de complexité intrinsèque : l'analyse est exponentielle en le nombre de composants du système. Dans ces travaux, le modèle est décrit dans sa vision compositionnelle comme illustré par la figure 3.1 page 65. Ce travail étant antérieur à celui sur les motifs, l'analyse est effectuée sur des fautes simples.

3.3.3.1 Diagnosticabilité locale

Le principe est d'analyser localement les composants afin d'en extraire les informations pertinentes, soit pour conclure directement, soit pour les propager vers l'analyse d'autres composants. Tout événement de faute f étant généralement associé à un phénomène physique identifié dans l'espace, ce travail part de l'hypothèse qu'une faute f ne peut se produire que sur un composant et un seul, idem pour les événements observables.

Définition 3.22 (Diagnosticabilité locale d'une faute f) *Soit γ un sous-système de langage $\mathcal{L}(\gamma)$ contenant le composant où la faute peut se manifester, la faute f est localement diagnosticable dans γ si*

$$\exists n \in \mathbb{N}, \forall \tau_1 \text{ une séquence } f\text{-concordante de } \mathcal{L}(\gamma), \forall \tau_2 \in \mathcal{L}(\gamma)/\tau_1 :$$
$$|obs(\tau_2)| \geq n \implies (\forall \tau_3 \in \mathcal{L}(\gamma), obs(\tau_3) = obs(\tau_1.\tau_2) \implies \tau_3 \text{ est } f\text{-concordante.})$$

La définition 3.22 est la version locale de la diagnosticabilité classique (définition 3.21) réduite aux motifs de fautes simples. [4] En particulier si le sous-système γ est le système lui-même, cette définition est équivalente. Il pourrait être intuitif de croire que si une faute est localement diagnosticable sur un sous-système γ alors elle est globalement diagnosticable, mais il n'en est rien si l'on n'impose pas une nouvelle condition sur le système : l'*observabilité vivante du sous système*. Un sous-système γ a une observabilité vivante si pour toute trajectoire du système global se terminant avec un événement observable issu de γ, il existe un nombre n tel que pour toute continuation de cette trajectoire dont la taille est supérieure à n, il existe au moins un événement observable issu de γ dans la continuation.

Théorème 3.1 ([34]) *Si une faute f est localement diagnosticable dans un sous-système γ alors f est diagnosticable dans le système si l'observabilité de γ est vivante dans le système.*

On peut interpréter ce résultat de deux façons différentes. La première est que si le système sous-jacent dispose *a priori* de la propriété de l'observabilité vivante locale, alors on dispose d'un moyen plus efficace d'analyser la diagnosticabilité du système en étudiant la diagnosticabilité locale de sous-systèmes. C'est par exemple le cas de systèmes à composants synchrones (tous les composants évoluent en même temps) sans cycles de non-observables. La deuxième interprétation est plus négative, à savoir qu'il suffit que le sous-système complémentaire de γ soit autorisé à évoluer indéfiniment (avec une infinité d'observations) et indépendamment de γ pour que la propriété de diagnosticabilité de f soit perdue au niveau du système. La condition de diagnosticabilité impose des conditions très restrictives sur les systèmes modélisés de façon compositionnelle.

3.3.3.2 Méthode d'analyse

Le principe de cette méthode est de considérer chaque faute f_i du système séparément et de déterminer par agrégations successives un sous-système γ dans lequel on a la garantie que la faute est localement diagnosticable. Lors des agrégations, l'objectif est de ne conserver que l'information qui est potentiellement à l'origine de

4. Il est à noter que l'extension de ces travaux aux motifs plus complexes est loin d'être triviale.

la non diagnosticabilité. La méthode est décrite par l'algorithme 3.5. Elle repose sur la création successive de vérificateurs [Yoo and Lafortune, 2002] (ou, plus précisément de sous-parties de vérificateurs). La construction d'un vérificateur peut être vue comme le *produit jumelé* d'une machine à états qui se nomme le *diagnostiqueur interactif*.

Soit $\gamma = \{\mathcal{A}_{\gamma_1}, \dots, \mathcal{A}_{\gamma_n}\}$ un sous-système, on note $I_i(\gamma)$ l'ensemble des événements interagissant entre composants de γ uniquement, $I_e(\gamma)$ l'ensemble des événements inter-agissants de γ également en interaction avec un composant extérieur à γ.

Définition 3.23 (Diagnostiqueur interactif) *Le diagnostiqueur interactif $\Delta_\gamma(f)$ de γ est l'automate (Q, Σ, T, q_0).*

- $q_0 = ((q_{\gamma_1}, \dots, q_{\gamma_n}), \emptyset)$ *est l'état initial où $q_{\gamma_j}, j \in \{1, \dots, n\}$ est l'état initial de $\mathcal{A}_{\gamma_j}$.*
- $\Sigma = \bigcup_{j=1}^{n} \Sigma_o^{\gamma_j} \cup I_e(\gamma)$.
- *Q est l'ensemble des états.*
- *T est l'ensemble des transitions.*
- $Q = \bigcup_{i=1}^{m} Q_i$ *et* $T = \bigcup_{i=1}^{m} T_i$ *où les T_i,Q_i,m sont définis comme suit.*

1. $Q_0 \longleftarrow \{q_0\}$, $T_0 = \emptyset$, $i = 0$.
2. *Tant que $Q_i \neq \emptyset$ ou $T_i \neq \emptyset$, répéter*
   - $Q_{i+1} \longleftarrow \emptyset, T_{i+1} \longleftarrow \emptyset$.
   - $\forall q = (q_\gamma, F) \in \bigcup_{j=0}^{i} Q_j$, *pour tout chemin de transitions* $q_\gamma \xrightarrow{e_1} \cdots \xrightarrow{e_k} q'_\gamma \xrightarrow{e} q''_\gamma$ *dans* $\mathcal{A}_{\gamma_1} \| \dots \| \mathcal{A}_{\gamma_n}$ *tel que* $e \in \Sigma$ *et aucun événement* $e_j, j \in \{1, \dots, k\}$ *n'est dans* Σ, *posons* $q'' = (q''_\gamma, F \cup \{f\})$ *si f est l'un des événements* $e_j, j \in \{1, \dots, k\}$ *ou* $q'' = (q''_\gamma, F)$ *sinon. Si* $q \xrightarrow{e} q''$ *n'est pas dans* $\bigcup_{j=0}^{i} T_j$ *alors* $T_{i+1} \longleftarrow T_{i+1} \cup \{q \xrightarrow{e} q''\}$. *Si q'' n'est pas dans* $\bigcup_{j=0}^{i} Q_j$ *alors* $Q_{i+1} \longleftarrow Q_{i+1} \cup \{q''\}$.
   - $i \longleftarrow i + 1$.
3. *m est tel que* $Q_{m+1} = \emptyset \wedge T_{m+1} = \emptyset \wedge (Q_m \neq \emptyset \vee T_m \neq \emptyset)$.

Le diagnostiqueur interactif est une machine à états qui ressemble au diagnostiqueur classique (voir par exemple la figure 3.6) mais avec les différences suivantes :

- elle est définie sur tout sous-système γ,
- les événements sur les transitions sont soit des observables de γ, soit des événements interactifs de $I_e(\gamma)$,
- la machine est non-déterministe, une seule hypothèse de diagnostic dans chaque état.

Intuitivement, le diagnostiqueur interactif permet de suivre le comportement observable de γ et les interactions potentielles qui en découlent avec les composants qui ne sont pas dans γ. Il propose également de poser des hypothèses dans chaque état sur l'occurrence ou non de la faute f si f est un événement qui a effectivement lieu dans γ. La notion de diagnostiqueur interactif $\Delta_\gamma(f)$ pour un sous-système γ dans lequel f

ne peut pas avoir lieu est également parfaitement définie. Dans ce dernier cas, chaque état de $\Delta_\gamma(f)$ est associé à $\emptyset$.

Le vérificateur de γ représente la confrontation des comportements du diagnostiqueur interactif qui ont la même projection observable [Yoo and Lafortune, 2002]. On construit pour cela le produit jumelé où seul les événements observables sont synchronisés. On note $\Delta^g_\gamma(f)$ le diagnostiqueur de gauche et $\Delta^d_\gamma(f)$ le diagnostiqueur de droite et (tout événement e non-observable de $\Delta_\gamma(f)$ est renommé $g : e$ dans $\Delta^g_\gamma(f)$, tout événement e non-observable de $\Delta_\gamma(f)$ est renommé $d : e$ dans $\Delta^d_\gamma(f)$).

Définition 3.24 (Vérificateur) *Soit $\gamma = \{\mathcal{A}_{\gamma_1}, \dots, \mathcal{A}_{\gamma_n}\}$ un sous-système le vérificateur de γ est l'automate :*

$$\mathcal{V}_\gamma(f) = \Delta^g_\gamma(f) \| \Delta^d_\gamma(f)$$

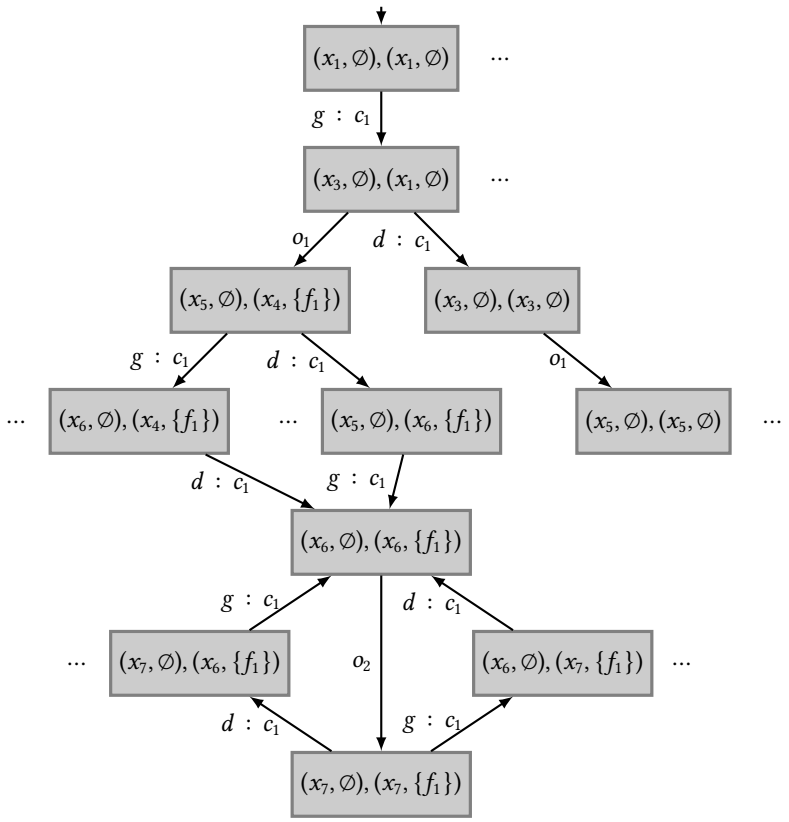


FIGURE 3.11 – Extrait du vérificateur de f_1 sur le sous-système $\gamma = \{\mathcal{A}_1\}$.

La figure 3.11 présente une partie du vérificateur du composant $\mathcal{A}_1$. Chaque état représente deux diagnostics possibles de γ pour deux trajectoires de γ émettant la

même séquence d'observation. Si la partie gauche et la partie droite de l'état ne s'accordent pas sur l'occurrence de la faute f, l'état du vérificateur est dit *ambigu*. L'analyse repose alors sur le résultat suivant :

Théorème 3.2 *La faute f est localement diagnosticable sur γ s'il n'existe pas de cycles observables d'états ambigus dans $\mathcal{V}_\gamma(f)$.*

On entend par cycle observable d'états ambigus un chemin de transitions menant à un cycle d'états ambigus et dans le cycle il y a au moins un événement observable. Il faut remarquer ici que le résultat n'est pas une équivalence. La faute f *peut* être localement diagnosticable même en présence de cycles observables d'états ambigus (dans ce cas cela signifie que les cycles en question ne sont pas globalement cohérents et ne se produisent donc jamais). Il peut également y avoir des cycles non observables, ces cycles ne sont également pas globalement cohérents dès lors que l'observabilité vivante de γ est garantie.

Entrées : $\{\mathcal{A}_1, \dots, \mathcal{A}_n\}$ modélisation compositionnelle du système
Entrées : f événement fautif sur le composant $\mathcal{A}_i$.
calcul vérificateur $\mathcal{V}_i(f)$;
$\mathcal{V}(f) \leftarrow$ supprimerNonAmbiguité($\mathcal{V}_i(f)$);
$\mathcal{C} \leftarrow \{\mathcal{A}_1, \dots, \mathcal{A}_n\} \setminus \{\mathcal{A}_i\}$;
tant que $\mathcal{V}(f) \neq \emptyset \wedge \mathcal{C} \neq \emptyset \wedge$ *nonDiag*($\mathcal{V}(f)$) **faire**
  Sélectionner $\mathcal{A}_j$ dans $\mathcal{C}$ communicant avec $\mathcal{V}(f)$;
  calcul vérificateur $\mathcal{V}_j(f)$;
  $\mathcal{V}(f) \leftarrow \mathcal{V}(f) \| \mathcal{V}_j(f)$;
  $\mathcal{V}(f) \leftarrow$ abstractionCommunications($\mathcal{V}(f)$);
  $\mathcal{V}(f) \leftarrow$ supprimerNonAmbiguité($\mathcal{V}(f)$);
fin tq
si $\mathcal{V}(f) = \emptyset$ **alors**
  Résultat : diagnosticabilité du sous-système de $\mathcal{V}(f)$
sinon
  Résultat : non diagnosticabilité de tout sur-système de $\mathcal{V}(f)$
fin si

Algorithme 3.5 : Analyse décentralisée de la diagnosticabilité.

Le principe de l'algorithme 3.5 est le suivant. L'objectif est de construire à chaque étape la sous-partie du vérificateur $\mathcal{V}_\gamma(f)$ qui peut conduire à un problème de diagnosticabilité. On procède d'abord en calculant le vérificateur de $\mathcal{V}_i(f)$ associé au composant $\mathcal{A}_i$ où f peut avoir lieu. On élimine de ce vérificateur toutes les parties non ambiguës. L'itération consiste à sélectionner un composant $\mathcal{A}_j$ qui interagit avec les composants du sous-système courant. On calcule son vérificateur $\mathcal{V}_j(f)$. Le vérificateur

$\mathcal{V}_j(f)$ s'appuie sur le diagnostiqueur interactif $\Delta_j(f)$ sans faute (tous les états sont du type $(q, \emptyset)$), il est ensuite synchronisé avec le vérificateur courant $\mathcal{V}$. Cette synchronisation va construire une machine qui va contenir des événements qui n'interagissent plus avec l'extérieur (ils interagissaient uniquement entre le sous-système précédent et le composant $\mathcal{A}_j$), ils sont supprimés par projection (abstractionCommunications). Cette synchronisation va également générer sur $\mathcal{V}$ des parties non-ambiguës qui sont supprimées ensuite (supprimerNonAmbiguité).

L'algorithme se termine dans deux cas. Le premier est quand $\mathcal{V}$ est vide, tout ce qui est sujet à la non-ambiguité, à la non-diagnosticabilité ayant été supprimé au cours des agrégations successives conduit à la conclusion que f est localement diagnosticable dans le sous-système de $\mathcal{V}$. Sous l'hypothèse que l'observabilité de γ est vivante dans le système, f est diagnosticable dans le système. Le deuxième cas est quand $\mathcal{V}$ n'est pas vide. L'algorithme s'est arrêté parce qu'il n'a pu trouver dans $\mathcal{C}$ un autre composant interagissant avec le sous-système de $\mathcal{V}$, par construction, $\mathcal{V}$ contient des cycles observables ambigus sans interaction avec les composants qui sont encore dans $\mathcal{C}$. Non seulement dans ce cas f n'est pas localement diagnosticable sur γ mais en plus, f n'est pas diagnosticable dans le système tout entier.

3.3.3.3 Exemple

Reprenant l'exemple du système de la figure 3.1, on constate que sur la figure du vérificateur de f_1 sur le composant $\mathcal{A}_1$ (figure 3.11), il y a des cycles d'états ambigus observables. L'algorithme 3.5 nécessite la synchronisation de ce vérificateur avec le vérificateur d'un composant interagissant. Dans cet exemple, les vérificateurs de chaque composant doivent être synchronisés pour conclure que f_1 n'est localement diagnosticable dans aucun sous-système (f_1 n'est pas diagnosticable dans le système entier).

Concernant la faute f_2, son vérificateur sur le composant $\mathcal{A}_2$ contient également des cycles d'états ambigus observables. L'algorithme 3.5 nécessite la synchronisation de ce vérificateur avec le vérificateur d'un composant interagissant. Si on choisit le composant $\mathcal{A}_3$, le vérificateur de f_2 résultant du sous-système $\gamma = \{\mathcal{A}_2, \mathcal{A}_3\}$ ne contient pas de cycles observables ambigus. La faute f_2 est localement diagnosticable dans $\gamma = \{\mathcal{A}_2, \mathcal{A}_3\}$. Dans l'exemple en question le sous-système $\gamma = \{\mathcal{A}_2, \mathcal{A}_3\}$ a une observabilité vivante dans le système, f_2 est donc diagnosticable au sens de la définition globale 3.21.

3.3.4 Analyse par agrégations distribuées

L'analyse présentée précédemment part du principe que l'on a l'hypothèse d'observabilité vivante déjà acquise pour ramener le problème de la diagnosticabilité globale à la détermination d'un sous-système γ dans lequel la faute f est localement diag-

nosticable. Dans le cadre de la thèse d'Anika Schumann, une extension de l'analyse précédente a été proposée afin de supprimer cette limitation [62]. Cette méthode repose sur la détection de *cycles observables d'états potentiellement ambigus* dans les vérificateurs $\mathcal{V}_{\gamma_j}$ obtenus par agrégations distribuées. Le principe est de construire le diagnostiqueur interactif $\Delta_i(f)$ de *chaque composant* $\mathcal{A}_i$ (définition 3.23) et d'obtenir ensuite leur vérificateur local $\mathcal{V}_i(f) = \Delta_i^g(f) \| \Delta_i^d(f)$ (définition 3.24) puis de les agréger de manière distribuée jusqu'à obtenir un critère de décision, à savoir l'absence de cycles observables d'états potentiellement ambigus.

Définition 3.25 (État potentiellement ambigu) *Soit $\mathcal{V}_\gamma(f)$ le vérificateur de f sur un sous-système γ quelconque, soit q un état de $\mathcal{V}_\gamma(f)$ il est* potentiellement ambigu *si l'une des conditions suivantes est vérifiée :*

- *f a lieu dans γ et q est ambigu dans $\mathcal{V}_\gamma(f)$;*
- *f n'a pas lieu dans γ.*

Définition 3.26 (Vérificateur réduit) *Soit $\mathcal{V}_\gamma(f)$ le vérificateur de f sur un sous-système γ quelconque, le vérificateur réduit $R(\mathcal{V}_\gamma(f))$ de $\mathcal{V}_\gamma(f)$ est la sous partie du système de transitions de $\mathcal{V}_\gamma(f)$ qui rassemble les séquences de transitions dont l'état cible est un état potentiellement ambigu.*

La méthode proposée dans [62] repose alors sur le résultat suivant.

Théorème 3.3 *Une faute f est diagnosticable ssi il existe un ensemble de vérificateurs réduits $\{R(\mathcal{V}_{\gamma_1}(f)), \ldots, R(\mathcal{V}_{\gamma_k}(f))\}$ tel que*

1. *$\{\gamma_1, \ldots, \gamma_k\}$ est une partition du système en sous-systèmes et f a lieu dans γ_1 ;*
2. *aucun des vérificateurs réduits $R(\mathcal{V}_{\gamma_j}(f))_{j\in\{1,\ldots,k\}}$ ne contient de cycles observables d'états potentiellement ambigus indépendants.*

Intuitivement, la faute f est diagnosticable si l'on arrive à déterminer une partition en sous-systèmes pour laquelle on ne peut établir pour chaque sous-système l'existence d'un cycle observable d'états potentiellement ambigus indépendants. Un cycle indépendant dans un vérificateur réduit $R(\mathcal{V}_{\gamma_j}(f))_{j\in\{1,\ldots,k\}}$ ne met en jeu que des événements internes de γ – issus uniquement de $I_i(\gamma)$ – autrement dit ce comportement est globalement cohérent, il existe effectivement dans le comportement global. Si ce cycle appartient au vérificateur du sous-système γ_f où la faute f a effectivement lieu, on a mis en évidence un comportement global infiniment observable et ambigu, la faute n'est donc pas diagnosticable. Si un tel cycle existe dans un autre sous-système $\gamma_{\bar{f}}$, alors on a démontré que l'observabilité de γ_f n'est pas vivante, il existe des comportements globaux pour lequel aucune observable de γ_f ne se produit, dès l'occurrence de la faute f dans γ_f, le diagnostic pourra rester définitivement ambigu, f n'est pas diagnosticable. La méthode est décrite par l'algorithme 3.6. On parle ici d'agrégation

distribuée car la stratégie de sélection des agrégations n'est pas nécessairement entre le vérificateur du sous-système où la faute a lieu et un autre comme dans l'analyse décentralisée de la section 3.3.3.2. Il peut s'agir de sélectionner deux vérificateurs où f n'a pas lieu. Le choix des agrégations se fait sur la nécessité que les deux vérificateurs interagissent effectivement l'un avec l'autre.

Entrées : $\{\mathcal{A}_1, \dots, \mathcal{A}_n\}$ modélisation compositionnelle du système
Entrées : f événement fautif sur le composant $\mathcal{A}_i$.
calcul des vérificateurs réduits $\pi \longleftarrow \{R(\mathcal{V}_j(f))\}_{j\in\{1,\dots,n\}}$;
tant que *il existe des cycles observables d'états potentiellement ambigus non indépendants dans* π *et sélection encore possible dans* π **faire**
  Sélection de $(R(\mathcal{V}_{j_1}(f)), R(\mathcal{V}_{j_2}(f)))$ dans π tel que $R(\mathcal{V}_{j_1}(f))$ interagit avec $R(\mathcal{V}_{j_2}(f))$;
  $\pi \longleftarrow \pi \cup \{R(R(\mathcal{V}_{j_1}(f))\|R(\mathcal{V}_{j_2}(f)))\} \setminus \{R(\mathcal{V}_{j_1}(f)), R(\mathcal{V}_{j_2}(f))\}$
fin tq
si *pas de observables d'états potentiellement ambigus indépendants* **alors**
  Résultat : diagnosticable
sinon
  Résultat : pas diagnosticable
fin si

Algorithme 3.6 : Analyse de la diagnosticabilité par agrégation distribuée.

3.3.5 Diagnosticabilité de motifs d'intérêt

Les travaux présentés précédemment, comme la plupart des travaux sur la diagnosticabilité de la communauté, se limitent à l'analyse de l'occurrence de fautes simples ou plus généralement à l'analyse d'une classe de fautes (un ensemble disjonctif de fautes simples $f_1 \vee f_2 \vee \cdots \vee f_n$) [Lin, 1994], [Sampath et al., 1995], [Jiang et al., 2001], [Yoo and Lafortune, 2002], [Cimatti et al., 2003], [Rintanen and Grastien, 2007], [Grastien, 2009], [Haar et al., 2009], [Cabasino et al., 2009], [Moreira et al., 2011], [Cabasino et al., 2012], [Basile et al., 2012], [Liu et al., 2014c], [Cabasino et al., 2014].

Dans le cadre de la thèse de Houssam-Eddine Gougam, l'objectif était d'analyser des comportements de système plus complexes que la simple occurrence d'événement. Ces comportements sont des motifs d'intérêt (voir la définition 3.9 page 61 qui est elle-même issue de ces travaux [25]). Dans la littérature, les travaux sur ce sujet sont peu nombreux contrairement au cas de la faute simple [Jiang and Kumar, 2004], [Jéron et al., 2006], [Ye and Dague, 2012]. Pourtant, l'extension aux motifs d'intérêt augmente l'expressivité des modèles traités et pose également de nouveaux problèmes dans la résolution. Les travaux décrits ici proposent une méthode effective pour l'analyse de la diagnosticabilité dans toute la généralité de la

définition 3.21 page 87. Cette méthode s'appuie sur une technique de vérification de modèle [Clarke et al., 1999]. Le formalisme utilisé dans cette étude est le *réseau de Petri étiqueté à priorité* [Berthomieu et al., 2007]. Ce choix a été guidé par deux raisons : la première est la possibilité de traiter la vérification de modèle décrit dans ce type de formalisme par l'outil TINA développé au sein du LAAS [Berthomieu et al., 2004], la seconde est de profiter de ce formalisme pour modéliser les motifs d'intérêt de façon la plus concise.

3.3.5.1 Modélisation du système

Le formalisme utilisé pour modéliser le système et les motifs d'intérêt est une extension des réseaux de Petri classique.

Définition 3.27 (RPEPL) *Un réseau de Petri étiqueté à priorité de type L (acronyme RPEPL) est un septuplet* $N = \langle P, T, A, \succ, \ell, \Sigma, Q \rangle$ *tel que :*

- P *est un ensemble fini de nœuds (places);*
- T *est un ensemble fini de nœuds (transitions) et* $P \cap T = \emptyset$*;*
- $A \subseteq (P \times T) \cup (T \times P)$ *est une relation binaire modélisant les arcs entre les places et les transitions;*
- *la relation de priorité* $\succ \subseteq T^2$ *est une relation binaire :*
  - *transitive* $(t_1 \succ t_2 \wedge t_2 \succ t_3 \Longrightarrow t_1 \succ t_3)$*;*
  - *non réflexive* $(\forall t \in T, t \not\succ t)$*;*
  - *et non symétrique* $(\forall t_1, t_2 \in T, t_1 \succ t_2 \Longrightarrow t_2 \not\succ t_1)$*;*
- Σ *est un ensemble fini d'étiquettes de transitions (événements);*
- $\ell : T \longrightarrow \Sigma$ *est la fonction d'étiquetage des transitions.*
- Q *est un ensemble de marquages accepteurs.*

L'état courant d'un RPEPL est un marquage, à savoir une fonction $M : P \longrightarrow \mathbb{N}$ qui associe à chaque place un nombre de jetons. Un RPEPL marqué est un couple $\Theta = \langle N, M_0 \rangle$ où M_0 est le marquage initial. L'ensemble précondition $\text{pre}(n)$ d'un nœud n est l'ensemble des nœuds $\text{pre}(n) = \{n' \in P \cup T : (n', n) \in A\}$ et l'ensemble postcondtion $\text{post}(n)$ d'un nœud n est l'ensemble des nœuds $\text{post}(n) = \{n' \in P \cup T : (n, n') \in A\}$. La relation $\succ$ définit un ensemble de priorités qui modifient les règles classiques de la sensibilisation des transitions du réseau. Une transition t est sensibilisée par un marquage M si et seulement si $(\forall p \in \text{pre}(t), M(p) > 0) \wedge (\forall t' \in T, t' \succ t \Longrightarrow \exists p' \in \text{pre}(t'), M(p') = 0)$. Cette règle de sensibilisation garantit que toute transition est sensibilisée dans le sens classique (lorsque $\succ = \emptyset$) mais évite de sensibiliser des transitions non-prioritaires par rapport à d'autres transitions sensibilisées. Déclencher une transition sensibilisée t à partir d'un marquage M conduit à un marquage $M' = M \setminus \text{pre}(t) \cup \text{post}(t)$, souvent noté $M \xrightarrow{t} M'$. Un marquage M est atteignable s'il existe une séquence de transitions

$M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \cdots \xrightarrow{t_k} M$, également noté $M_0 \xrightarrow{s} M$ avec $s = t_1 \dots t_k$. L'ensemble des marquages atteignables est noté $R(\Theta, M_0)$.

Un RPEPL contient également une fonction d'étiquetage ℓ qui associe un événement (au sens de la définition 3.1) à une ou plusieurs transitions du réseau. Enfin, un RPEPL contient un ensemble Q de marquages accepteurs. Cet ensemble Q permet de définir le langage de type L, au sens de [Peterson, 1977], généré par le RPEPL qui est défini de la façon suivante :

Définition 3.28 (Langage de RPEPL) *Le langage généré par le RPEPL* $\Theta = \langle P, T, A, \succ, \ell, \Sigma, Q, M_0 \rangle$ *est :*

$$\mathcal{L}(\Theta) = \{\ell(r) \,:\, r \in T^* \wedge M_0 \xrightarrow{r} M \wedge M \in Q\}.$$

Dans la méthode proposée dans [25], le système est donc représenté par un RPEPL.

Définition 3.29 (Modèle RPEPL du système) *Le modèle du système est un RPEPL* $\Theta = \langle P, T, A, \succ, \ell, \Sigma, Q, M_0 \rangle$ *tel que :*

- *le réseau est borné* ($\exists n \in \mathbb{N}^+ \,:\, \forall M \in R(\Theta, M_0), \forall p \in P, M(p) \leq n$)*;*
- $Q = R(\Theta, M_0)$.

Avec la définition 3.29, le système à événements discrets *est représenté* par un RPEPL comme il peut être représente à l'aide d'automates dans une modélisation compositionnelle. Néanmoins, il s'agit toujours du même type de système, celui de la définition 3.4. Ici le langage $\mathcal{S}$ du système de modèle RPEPL Θ est tel que :

$$\mathcal{S} = \mathcal{L}(\Theta).$$

La figure 3.12 représente la modélisation d'un système manufacturier paramétrable de n lignes de production, chaque ligne étant composée de k opérations et pouvant recevoir m ressources à traiter en même temps. Ce modèle est une version modifiée du cas d'étude proposé dans la conférence internationale WODES'08 [Giua, 2007]. Le système est observé à l'aide du masque obs tel que $obs(w) = \{w\}$, $obs(t_s) = \{t_s\}$,$obs(t_e) = \{t_e\}$, $obs(t_{i,k+1}) = \{t_{i,k+1}\}, i \in \{2, \dots, n\}$, les autres événements étant non-observables. Ce système contient également un ensemble d'événements de fautes $f_1, \dots, f_{n-1}$.

3.3.5.2 Modélisation des motifs d'intérêt

L'objectif est ici de proposer une représentation de comportements d'intérêt plus sophistiqués que la simple occurrence d'un événement de faute. Un motif d'intérêt est défini comme un langage, on en propose ici une représentation sous la forme d'un RPEPL particulier.

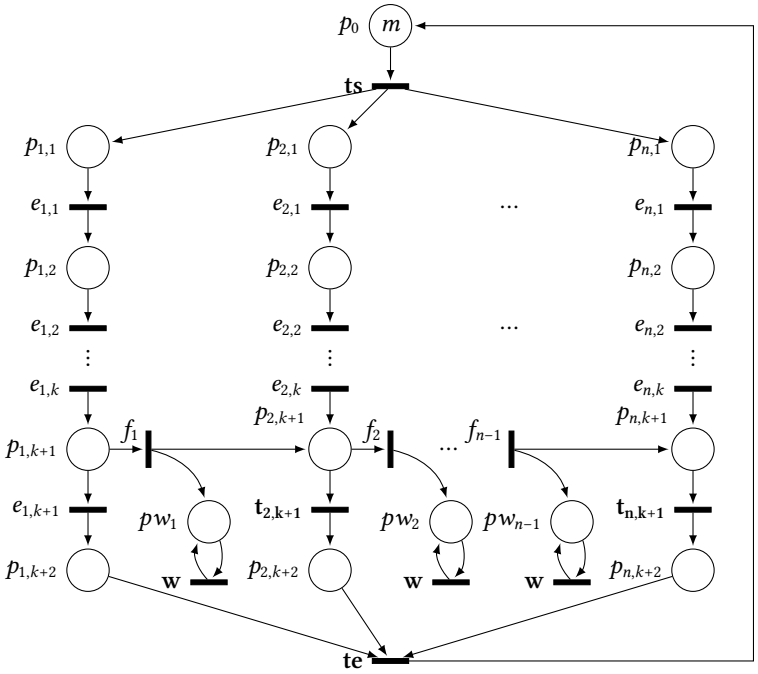


FIGURE 3.12 – Le système exemple de WODES'08 en sa version vivante.

Définition 3.30 (Modèle de motif) *Un motif* Ω *est un RPEPL*

$$\Omega = \langle P, T, A, \succ, \ell, \Sigma, Q, M_0 \rangle$$

tel que

1. *(initialisation)* $M_0 \notin Q$ *le langage du motif ne contient pas* ε *;*
2. *(bien-formé) de tout marquage atteignable* M *de* $R(\Omega, M_0)$ *il existe une séquence de transitions* $r \in T^*$ *telle que* $M \xrightarrow{r} M'$ *et* $M' \in Q$ *;*
3. *(déterminisme des événements) de tout marquage atteignable* M *de* $R(\Omega, M_0)$ *il n'y a pas d'événement* $e \in \Sigma$ *qui étiquette plus d'une transition sensibilisée par le marquage ;*
4. *(stabilité)* $\forall M \in Q, \forall M' : (\exists t \in T, M \xrightarrow{t} M') \Longrightarrow M' \in Q$.

Les conditions imposées servent à définir des motifs d'intérêt sensés (ils ne contiennent pas la séquence vide) mais également une structure de réseau qui soit compatible avec le traitement du motif pour l'analyse de la diagnosticabilité. Notamment, on interdit la présence de marquages atteignables et non accepteurs à partir desquels on ne peut aboutir à un marquage accepteur.

La figure 3.13 illustre un tel motif sur le système présenté sur la figure 3.12. À l'opposé du comportement de la faute classique, le motif $\Omega_1(k, l)$ illustre un comportement

d'intérêt plus sophistiqué, à savoir l réalisations normales (sans faute f_1) de la ligne de production 1 de longueur k.

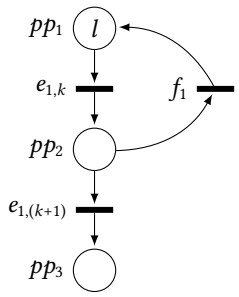


FIGURE 3.13 – Motif $\Omega_1(k, l)$ with $Q_1 = \{M : M(pp_3) \geq l\}$.

Afin d'illustrer le fait que les motifs d'intérêt étendent le problème de base, la figure 3.14 présente un motif $\Omega_2(n)$ exprimant la classe de faute $\{f_1, \ldots, f_n\}$: l'occurrence d'une faute parmi les n. Le traitement des motifs décrits dans l'analyse de diagnosticabilité est identique. Enfin, en modifiant l'ensemble des marquages accepteurs de $\Omega_2(n)$ on peut représenter également l'occurrence des n fautes (fautes multiples) qui est un comportement plus spécifique qui celui de $\Omega_2(n)$ et qui peut donc avoir une diagnosticabilité différente : on peut très bien ne pas pouvoir diagnostiquer avec certitude que l'une des fautes a effectivement eu lieu, mais pouvoir diagnostiquer avec certitude l'occurrence de ces mêmes fautes à chaque fois qu'elles ont toutes eu lieu.

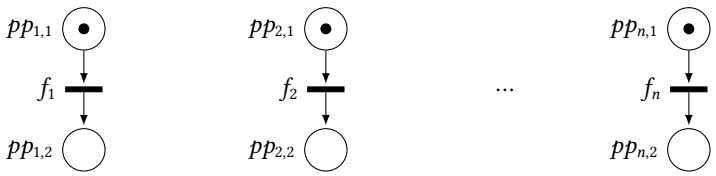


FIGURE 3.14 – Motif de la classe de fautes simples (1 parmi n) $\Omega_2(n)$ avec $Q_2 = \{M : \exists i \in \{1, \ldots, n\}, M(pp_{i,2}) \geq 1\}$. Motif de la faute multiple $\Omega_3(n)$ (n sur n), avec $Q_3 = \{M : \forall i \in \{1, \ldots, n\}, M(pp_{i,2}) \geq 1\}$.

En représentant les motifs de façon totalement indépendante du système, on met bien en évidence la distinction entre le modèle de comportement du système Θ et l'objectif de diagnostic Ω comme préconisé dans [Lamperti and Zanella, 2006].

3.3.5.3 Problème et méthode d'analyse

Le problème que l'on cherche à résoudre est celui de la diagnosticabilité de motifs au sens de la définition 3.21. Un motif μ est ici représenté par un RPEPL Ω tel que $\mathcal{L}(\Omega) =$

μ.[5] On va s'appuyer sur cette représentation pour résoudre ce problème. Comme pour toute méthode d'analyse de diagnosticabilité dans les SED, on élimine les cas triviaux de non-diagnosticabilité de système en faisant l'hypothèse suivante : le système ne génère pas de cycles d'événements non-observables.

La figure 3.15 présente le principe général de l'analyse de la diagnosticabilité d'un motif Ω sur un système Θ.

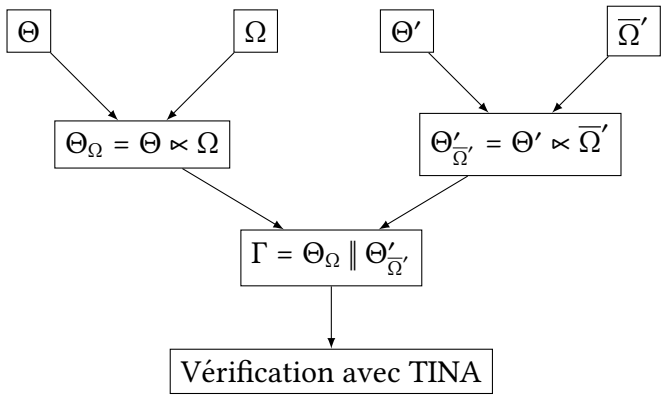


FIGURE 3.15 – Principe général de la méthode d'analyse de diagnosticabilité d'un motif Ω sur un système Θ.

La méthode repose sur la définition de deux opérations. Le première et la plus importante est le produit motif-système $\Theta \propto \Omega$. Ce produit est original et spécifique au problème de la concordance d'un système sur un motif.

Définition 3.31 (Produit motif-système) *Soit* $\Theta_1 = \langle P_1, T_1, A_1, \succ_1, \ell_1, \Sigma_1, Q_1, M_{10}\rangle$, $\Theta_2 = \langle P_2, T_2, A_2, \succ_2, \ell_2, \Sigma_2, Q_2, M_{20}\rangle$ *deux RPEPLs tels que* $P_1 \cap P_2 = \emptyset$, $T_1 \cap T_2 = \emptyset$ *et* $\Sigma_2 \subseteq \Sigma_1$, *le produit* $\Theta_1 \propto \Theta_2$ *est le RPEPL* $\langle P, T, A, \succ, \ell, \Sigma, Q, M_0\rangle$ *défini comme suit.*

- $P = P_1 \cup P_2$.
- $T = T_1 \cup T_s$ *tel que*
  - $T_s = \bigcup_{l\in\Sigma_1\cap\Sigma_2}\{t_1\|t_2 : \exists t_1 \in T_1 \wedge \exists t_2 \in T_2 \wedge \ell_1(t_1) = \ell_2(t_2) = l\}$ *est l'ensemble des transitions synchronisées, la transition* $t_1\|t_2$ *résultant de la synchronisation de* $t_1 \in T_1$ *et de* $t_2 \in T_2$.
- $A = A_1 \cup A_s$ *tel que*

$$\begin{aligned} A_s = &\{(p,t) : p \in P_1, t = t_1\|t_2, (p,t_1) \in A_1\} \\ &\cup \{(p,t) : p \in P_2, t = t_1\|t_2, (p,t_2) \in A_2\} \\ &\cup \{(t,p) : p \in P_1, t = t_1\|t_2, (t_1,p) \in A_1\} \\ &\cup \{(t,p) : p \in P_2, t = t_1\|t_2, (t_2,p) \in A_2\}. \end{aligned}$$

5. Ceci n'est pas tout à fait vrai. μ est en fait la restriction de $\mathcal{L}(\Omega)$ aux mots dont aucun préfixe n'est dans $\mathcal{L}(\Omega)$. Ceci est due à la condition de stabilité dont la présence permet une modélisation des motifs plus aisés qui n'a aucune conséquence sur la diagnosticabilité de μ.

— $\succ = \succ_1 \cup \succ_s$, *avec :*

$$\succ_s = \{(t_s, t_1) \,:\, t_s = t_1 \| t_2 \in T_s\} \cup \bigcup_{t_s = t_1 \| t_2 \in T_s} \{ \bigcup_{t_i \in \{t_1, t_2\}} \{(t_s, t) \,:\, (t_i, t) \in \succ_i\} \cup \bigcup_{t_i \in \{t_1, t_2\}} \{(t, t_s) \,:\, (t, t_i) \in \succ_i\}\}.$$

— $\forall t = t_1 || t_2 \in T_s, \ell(t) = \ell_1(t_1), \forall t \in T \cap T_1, \ell(t) = \ell_1(t), \forall t \in T \cap T_2, \ell(t) = \ell_2(t)$.
— $\Sigma = \Sigma_1 \cup \Sigma_2$.
— $Q = \{q_1 \cup q_2 \,:\, (q_1, q_2) \in Q_1 \times Q_2\}$.
— $M_0(p) = M_{10} \cup M_{20}$.

Grâce au système de priorités disponibles dans les RPEPLs, le produit $\Theta \ltimes \Omega$ confronte le comportement de Θ avec celui de Ω. À chaque fois qu'un événement de Θ peut se synchroniser avec le même événement dans Ω, la synchronisation a effectivement lieu, sinon seul l'événement de Θ se produit. La définition du produit $\Theta \ltimes \Omega$ mène au résultat suivant.

Théorème 3.4

$$\mathcal{L}(\Theta \ltimes \Omega) = \{\tau \in \mathcal{L}(\Theta) \,:\, \tau \textit{ est } \mathcal{L}(\Omega)\textit{-concordante}\}.$$

Cette opération est également appliquée sur la *négation* $\overline{\Omega}$ du motif Ω. $\overline{\Omega}$ ne diffère de Ω que par ses marquages accepteurs (les marquages accepteurs de $\overline{\Omega}$ étant le complément des marquages accepteurs de Ω dans l'ensemble des marquages atteignables de Ω), ce qui mène au résultat suivant :

Théorème 3.5

$$\mathcal{L}(\Theta \ltimes \overline{\Omega}) = \{\tau \in \mathcal{L}(\Theta) \,:\, \tau \textit{ n'est pas } \mathcal{L}(\Omega)\textit{-concordante}\}.$$

La prochaine étape consiste alors à confronter $\Theta \ltimes \Omega$ et $\Theta \ltimes \overline{\Omega}$ pour caractériser les comportements observables de Θ pouvant être expliqués à la fois par un comportement de Θ concordant et un comportement de Θ non-concordant. C'est le principe du produit jumelé adapté de [Jiang et al., 2001],[Yoo and Lafortune, 2002] qui s'obtient en synchronisant classiquement les transitions des deux réseaux ayant les mêmes étiquettes observables :

$$\Gamma = \Theta_\Omega \parallel \Theta'_{\overline{\Omega}'}$$

où $\Theta_\Omega = \Theta \ltimes \Omega$ et $\Theta'_{\overline{\Omega}'}$ est un simple renommage des événements non-observables de $\Theta \ltimes \overline{\Omega}$.

Le RPEPL Γ est construit ainsi à l'aide de deux clones de Θ. Chaque marquage atteignable de Γ représente deux marquages atteignables de Θ. Si l'un des marquages contient un marquage accepteur de Ω et que l'autre contient un marquage accepteur de $\overline{\Omega}$ alors le marquage de Γ est ambigu. Le réseau Γ est finalement associé au résultat suivant :

Théorème 3.6 *Un motif Ω est diagnosticable sur Θ ssi Γ ne contient pas de cycles ambigus.*

La dernière étape consiste alors à vérifier l'existence de tels cycles à l'aide du vérificateur de modèle TINA dont l'une des particularités est sa capacité à analyser des réseaux de Petri à priorité.

3.3.5.4 Résultats expérimentaux

La table 3.5 présente quelques résultats d'analyse du système de la figure 3.12 pour le motif de la figure 3.13. Ces résultats ont été obtenus en calculant dans un premier temps la structure de Kripke $K(\Gamma_1)$ de Γ_1 qui est un système de transitions associant à tous les marquages atteignables de Γ_1 l'ensemble des propriétés atomiques (des contraintes sur les places) qui les satisfont [Kripke, 1963]. Dans un deuxième temps, le vérificateur de propriétés dédiés de TINA (`selt`) est utilisé pour vérifier une propriété de type SE-LTL exprimant la non-existence de cycles ambigus [Berthomieu et al., 2007]. Cette propriété s'exprime selon le principe suivant :

$$\varphi_{\text{DIAG}} = \forall\Box((M \in Q \wedge M \in \overline{Q}) \Longrightarrow (\Diamond(M' \notin \overline{Q} \vee \textbf{dead})))$$

Intuitivement, φ_{DIAG} exprime que pour toute trajectoire de Γ ($\forall$), il est toujours vrai ($\Box$) que, dès lors que le marquage courant est ambigu ($(M \in Q \wedge M \in \overline{Q}) \Longrightarrow$), en un nombre fini de transitions ($\Diamond$), la trajectoire conduira à un blocage (**dead**) ou à un marquage non-ambigu ($M' \notin \overline{Q}$).

3.3.6 Extensions

Les travaux présentés dans la section précédente sont également liés à des travaux annexes. En particulier, toujours dans le cadre de la thèse de Houssam-Eddine Gougam, des travaux sur l'analyse de diagnosticabilité de motifs par dépliage de réseaux de Petri [McMillan and Probst, 1995, Esparza et al., 2002] ont été proposés dans [25, 24] et une version se restreignant à une analyse de discriminabilité de motifs a été proposée dans [26].

La question de la diagnosticabilité qui a été traitée précédemment est binaire, à savoir : *le système est-il diagnosticable ou non ?* La réponse à cette question a des conséquences totalement différentes en fonction du type de la réponse. Si le système

m	n	k	$nS(K(\Gamma_1))$	$nT(K(\Gamma_1))$	Temps(s)	Résultat
1	2	1	359	942	0.004	Yes
1	2	2	1094	3246	0.004	Yes
1	2	3	2619	8314	0.012	Yes
1	2	4	5366	17778	0.032	Yes
1	3	1	5678	26415	0.036	No
1	3	2	28093	139800	0.196	No
1	3	3	97718	503899	0.752	No
1	3	4	272537	1437600	2.056	No
1	4	1	113835	886663	1.28	No
1	4	2	898024	7124005	10.86	No
1	4	3	4476103	35592707	55.848	No
1	4	4	16776780	133151809	211.032	No
1	5	1	2617158	32273068	50.224	No
1	5	2	32768435	398600857	840.872	No
2	2	1	9884	42408	0.072	No
2	2	2	77689	390460	0.672	No
2	3	1	1083142	8472878	14.572	No
2	3	2	24770740	218976350	405.012	No
3	2	1	126926	694178	1.24	No
3	3	1	51727198	510135126	1256.88	No

TABLE 3.5 – Diagnosticabilité du motif $\Omega_1(k, 3)$.

est diagnosticable, alors on sait que l'on peut mettre en place un système de diagnostiqueurs toujours capable de diagnostiquer avec certitude l'occurrence d'une faute (ou plus généralement d'un motif). Ainsi si la réponse est affirmative, le problème de diagnostic est résolvable.

À l'inverse, dans le cas d'une réponse négative, on sait que l'on a mis en évidence un comportement du système pour lequel le système n'est pas diagnosticable, mais c'est bien tout. Autant la réponse affirmative est constructive, autant la réponse négative ne l'est pas. Que peut-on faire si le système n'est pas diagnosticable ?

La raison principale de la non-diagnosticabilité d'un système est généralement lié au trop faible nombre de capteurs qui l'instrumentent. La simple réponse *non* n'offre aucun moyen d'améliorer la situation. Dans le cas d'une réponse négative, une réponse plus informative est nécessaire afin d'assister les concepteurs du système à de potentiels changements.

À travers mes travaux ce problème de retour d'information en cas de réponse négative prend plusieurs formes.

3.3.6.1 Assistance pour la conception d'un système diagnosticable

Contrairement à la plupart des contributions sur l'analyse de la diagnosticabilité qui se contentent d'une simple réponse négative dès lors que le système est prouvé non-diagnosticable, les méthodes qui ont été développées et décrites ci-dessus ont été étendues afin de générer des informations plus constructives. La contribution [36] propose notamment de transformer le problème initial de l'analyse de la diagnosticabilité en la recherche de *scénarios non diagnosticables* dans un sous-système γ donné. Considérant par exemple l'analyse décentralisée (voir la section 3.3.3), un scénario non diagnosticable sur un sous-système γ est représenté par un cycle ambigu dans le vérificateur $\mathcal{V}(f)$: il s'agit de la confrontation de deux comportements différents, l'un fautif et l'autre non, ayant des comportements observables identiques et non bornés. Dans [36], on propose ainsi de retourner comme réponse négative cet ensemble de scénarios non-diagnosticables pour un sous-système donné. Ce type d'information a plusieurs conséquences.

1. De l'énumération des scénarios non-diagnosticables il est possible de déduire pour un sous-système donné quels sont les scénarios diagnosticables sur lesquels on a la garantie effective de pouvoir conclure avec certitude.
2. La caractérisation des scénarios non-diagnosticables est la première étape pour déterminer de nouveaux placements de capteurs afin de les éliminer.

Un autre point souvent omis dans la littérature est lié à la complexité des algorithmes d'analyse de diagnosticabilité. Cette analyse est certes polynomiale en le nombre d'états du modèle global (pour les automates) ou du graphe des marquages (pour les réseaux de Petri), mais elle est donc également exponentielle en le nombre

de composants (pour les automates) ou le nombre de places (pour les réseaux de Petri). D'un point de vue pratique, l'analyse de diagnosticabilité est ainsi limitée par les ressources informatiques disponibles. Les méthodes décentralisées et distribuées [34], [36], [62] proposent de faire face à cette difficulté en ayant tout de même une réponse dès lors que la capacité maximale des ressources informatiques est atteinte. Dans [34], [36], [62], si l'on est confronté à l'impossibilité d'agréger un composant avec le sous-système courant, on peut néanmoins établir les conclusions partielles suivantes :

- Tout scénario qui n'est pas inclus dans le vérificateur réduit est diagnosticable dans le sous-système courant.
- Si le système est diagnosticable, alors il est nécessaire, sauf cas particuliers, de mettre en place un diagnostiqueur qui observe un sous-système incluant le sous-système courant. Il est possible que le sous-système courant soit diagnosticable si tous les comportements ambigus contenus dans le vérificateur courant ne sont pas globalement admissibles [36].

3.3.6.2 Caractérisation de prérequis pour la diagnosticabilité

L'analyse de la diagnosticabilité n'est pas une fin en soi. Cette analyse permet ensuite de décider de l'architecture pour le diagnostic du système analysé, ou bien d'un retour sur conception du système dès que le coût de ce retour est économiquement viable [53], ce qui est particulièrement le cas dans la maintenance des systèmes aéronautiques. Une contribution issue de la thèse de Pauline Ribot développe cet aspect de retour sur conception pour l'amélioration de la diagnosticabilité. L'objectif a été de profiter de l'analyse décentralisée issue de [34] et de [36] pour proposer un algorithme décentralisé de résolution de placements de capteurs [Narasimhan et al., 1998], [Debouk et al., 1999], [Travé-Massuyès et al., 2001], [Debouk et al., 2002b], [Cassez et al., 2007], [Torta and Torasso, 2007], [Thorsley and Teneketzis, 2007] qui est posé ici comme un problème d'optimisation de coût [56].

Afin de mettre en œuvre toute une architecture de diagnostic qui peut garantir l'isolation d'un ensemble de fautes F, il y a deux problèmes à résoudre :

- comment placer les capteurs sur le système (*instrumentation*) ;
- comment définir l'architecture logicielle pour la surveillance et le diagnostic du système (*déploiement*).

Chaque problème est lié à un coût. Le coût C_I^i est le coût de l'instrumentation du composant i. Le coût C_A^i est le coût de l'architecture logicielle (diagnostiqueurs, gestion des communications capteurs, calculateurs) devant être déployée sur le composant i. Ainsi le coût global minimal est :

$$C_G = min_{\{A,I\}} \sum_{i=1}^{p} (C_I^i + C_A^i)$$

où $\{A, I\}$ représente l'ensemble des instrumentations et architectures possibles sur

le système en question, système qui est composé de p composants. L'algorithme 3.7 part de ce principe et propose une méthodologie pour établir le meilleur compromis afin de placer des capteurs pour rendre une faute simple f localement diagnosticable (au sens de la définition 3.22) sur le système tout en minimisant le coût de son instrumentation (coût des capteurs et de leur placement) et de l'architecture logicielle associée (réseau de communication, calculateurs, complexité algorithmique).

Entrées : Modèle compositionnel : $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$
Entrées : μ un motif local sur un composant $\mathcal{A}_i$
Entrées : Masque observable initial : obs
$C_G^0 \longleftarrow \infty$;
$k \longleftarrow 1$;
répéter
  $C_G^k \longleftarrow \infty$;
  pour tous les $\gamma \in SousSystèmes(\mathcal{A}_i, k)$ **faire**
    $C_S \longleftarrow CoûtSurveillance(\gamma, obs)$;
    $C_D \longleftarrow 0$;
    si $\neg$ *DiagnosticabilitéLocale*(γ, μ, obs) **alors**
      si $\neg$ *SolutionMaximale*$(\gamma, \mu) = \emptyset$ **alors**
        $(obs', C_D) \longleftarrow PlacerCapteurMinimal(\gamma, \mu, obs)$;
      fin si
    fin si
    $C_G^\gamma \longleftarrow C_S + C_D$;
    si $C_G^\gamma < C_G^k$ **alors**
      $C_G^k \longleftarrow C_G^\gamma$;
      $obs^k \longleftarrow obs'$;
    fin si
  fin pour tous
  $k \longleftarrow k + 1$;
jusqu'à $(C_G^{k-1} \geq C_G^{k-2}) \wedge (k \geq 2) \wedge (k \leq n)$;
retourner C_G^{k-2}, obs^{k-2}

Algorithme 3.7 : Algorithme décentralisé pour le placement de capteurs.

Le principe est d'opérer par agrégations successives de composants en interaction directe ou indirecte avec le composant où la faute peut se produire. On étudie tous les sous-systèmes possibles d'une taille donnée k en ordre croissant sur les k. L'étude s'arrête lorsque l'on a établi que le coût pour rendre diagnosticable un sous-système de taille k est inférieur à celui pour rendre diagnosticable un sous-système de taille $k + 1$. Ce critère d'arrêt s'appuie sur le fait que tout sous-système de taille supérieure à $k + 1$ ne pourra pas être associé à un coût inférieur à celui du sous-système de taille

k du fait en particulier de la complexité algorithmique exponentielle en le nombre de composants. Pour chaque sous-système, on étudie ainsi sa diagnosticabilité locale en utilisant par exemple la méthode proposée dans la section 3.3.3.2. Si le sous-système en question est localement diagnosticable, il est inutile d'ajouter des capteurs, le coût est alors réduit à la mise en place d'une architecture de diagnostic sur ce sous système (coût de la surveillance C_S). Si, au contraire, le sous-système en question n'est pas localement diagnosticable, alors on doit opérer un placement de capteurs (*PlacerCapteurMinimal*). Le principe du placement de capteur consiste à modifier le masque observable obs en obs' afin de garantir la diagnosticabilité locale. L'algorithme de placement de capteur est en soi issu de la littérature ([Debouk et al., 2002b],[Cassez et al., 2007]). Il est tout à fait possible qu'il n'existe pas de solution pour un sous-système donné, à savoir que le fait de rendre observable tout événement du sous-système en question qui n'est pas la faute f ne permet pas de rendre la faute diagnosticable sur ce sous-système.

3.3.7 Résumé

L'analyse de diagnosticabilité sur les systèmes à événements discrets est un problème crucial en amont de la mise en place d'un système de diagnostic car cette analyse détermine les capacités du système de diagnostic sur le système analysé. Les contributions présentées ici ont été développées à travers différentes collaborations et encadrements de thèse. Les premières contributions se sont orientées vers des analyses décentralisées et distribuées de la diagnosticabilité afin de mieux appréhender l'explosion combinatoire des analyses centralisées qui ont été développées dans les travaux antérieurs de la littérature mais également de mieux comprendre les problèmes de diagnosticabilité issus d'un système dont la représentation est une modélisation compositionnelle. La diagnosticabilité étant une propriété forte, elle impose des conditions fortes sur les interactions entre composants et sur leur observabilité. La deuxième contribution étend le problème à des motifs, c'est à dire des comportements plus complexes. L'analyse de la diagnosticabilité de motifs impose une difficulté supplémentaire, la localité du motif peut ne pas être sur un seul composant mais sur un groupe de composants, voire tous les composants du système, ce qui est un frein à l'analyse décentralisée. Nous avons donc opté pour cette deuxième contribution pour l'analyse par des techniques de vérification de modèles en s'appuyant sur des outils qui sont spécialisés dans la gestion de l'explosion combinatoire dans le calcul des graphes de marquages associés aux réseaux de Petri analysés (réduction d'ordre partiel, techniques des ensembles persistants,...). Un dernier apport à l'analyse de la diagnosticabilité est le type d'informations retourné par cette analyse dans le cas où le système n'est pas diagnosticable. Les analyses décentralisées/distribuées peuvent apporter une information plus riche, notamment sur la façon de placer de nouveaux capteurs afin d'améliorer l'état général de la diagnosticabilité du système.

3.4 Précision du diagnostic dans le cadre distribué

La section précédente présente les contributions sur l'analyse d'une propriété importante pour le diagnostic d'un système – la diagnosticabilité – propriété qui garantit que le diagnostic pourra conclure avec certitude et en un temps fini sur la présence effective d'une faute, d'un motif. Cette même section a présenté en particulier une analyse décentralisée qui a mis en avant les conditions très restrictives que le système doit mettre en œuvre pour qu'il soit effectivement diagnosticable et parmi ces conditions, certaines sont propres aux interactions entre un sous-système et le reste du système. Les contributions présentées dans cette section partent de ce constat : les conditions nécessaires à la diagnosticabilité globale du système imposent des conditions locales sur les interactions entre composants.

La complexité inhérente du problème du diagnostic impose qu'il n'est pas possible dans les cas réalistes de s'appuyer sur un modèle global pour effectuer le diagnostic, une approche décentralisée [Debouk et al., 2002a], [41] ou distribuée [Sengupta, 1998], [Benveniste et al., 2003], [Fabre et al., 2005], [Su and Wonham, 2005] est nécessaire avec pour objectif de n'établir qu'un diagnostic local qui soit *suffisant* et ne nécessite donc en aucun cas le calcul de la cohérence globale du système qui implique tous les composants du système. C'est l'objectif de la propriété de *précision* sur les systèmes à événements discrets introduite dans [48]. Peu importe que le système soit globalement diagnosticable ou pas, existe-t-il des sous-systèmes, ou encore des sous-configurations, pour lesquelles le diagnostic local de ces sous-systèmes est aussi précis que celui que l'on aurait obtenu si on avait établi le diagnostic global ?

3.4.1 Contexte des travaux

Le travail sur la précision des systèmes à événements discrets a été développé en parallèle des travaux sur la diagnosticabilité et les liens entre ces deux propriétés ont été établis. Le travail a été initié au cours du stage de Master 1 de Dimitry Kamenetsky (Australian National University) puis a été étendu dans le cadre d'une collaboration entre Alban Grastien (chercheur NICTA, Australie), Priscilla Kan John (doctorante ANU, Australie) et Pauline Ribot (doctorante sous ma direction). Dans ces travaux, nous reprenons le cadre usuel de la modélisation compositionnelle d'un système à événements discrets (pour plus de détails, voir la section 3.2.2.1).

3.4.2 Diagnostiqueur spécialisé

Nous avons vu que l'un des problèmes majeurs du diagnostic de systèmes à événements discrets est l'explosion combinatoire qui est exponentiel en le nombre de composants. On parle ici du problème général qui consiste à diagnostiquer l'*état global* du système et la présence de *tous les motifs d'intérêt*. Une façon de casser la complexité

du problème est de le simplifier. La *spécialisation du diagnostiqueur*, introduite dans [48], consiste à synthétiser un diagnostiqueur qui a la charge de n'identifier qu'un type de motif et un seul. Reprenant la définition 3.14 du diagnostic candidat, le diagnostiqueur spécialisé ne cherche qu'à savoir si un motif donné M est dans un diagnostic candidat ou pas. Dans le cas précis de cette étude, les motifs d'intérêt sont limités à la simple occurrence de fautes $\{F_1, \ldots, F_n\}$. [6]

Définition 3.32 (Diagnostiqueur spécialisé) *Un diagnostiqueur spécialisé de F est une fonction du problème* $(\mathsf{DS}, \mathcal{M}, \mathsf{OBS})$ *vers* { *F-présent, F-absent, F-ambigu* }.

Cette première définition est très générale. Elle indique seulement la finalité d'un diagnostiqueur spécialisé : pour une séquence d'observations donnée OBS d'un problème de diagnostic, le diagnostiqueur spécialisé retourne une modalité parmi trois :

— F-présent : l'événement F a effectivement eu lieu ;
— F-absent : l'événement F n'a pas eu lieu ;
— F-ambigu : l'événement F a peut-être eu lieu.

Dans [48], on se propose d'associer à chaque faute un tel diagnostiqueur et de ne construire ce diagnostiqueur que sur un sous-système γ. Ici, par diagnostiqueur, nous entendons une fonction qui peut-être mise en œuvre par n'importe quel moyen, notamment, il peut s'agir de l'un des algorithmes proposés en section 3.2.

Définition 3.33 (Diagnostiqueur spécialisé de F sur γ) *Soit γ un sous-système dans lequel F peut avoir lieu, un diagnostiqueur spécialisé de F sur γ (noté $\Delta_\gamma(F)$) est un diagnostiqueur spécialisé de F n'observant que les événements issus de γ et ne connaissant que le modèle des composants de γ.*

Le principe d'un algorithme de diagnostic par spécialisation est résumé par l'algorithme 3.8. Pour chaque faute F, on sélectionne un sous-système γ_F et on met en œuvre le diagnostiqueur spécialisé de F : $\Delta_{\gamma_F}(F)$. Le principe ensuite est de projeter les observations de OBS sur chacun des sous-systèmes γ_F et de retourner le résultat du diagnostiqueur spécialisé correspondant :

$$\Delta_{\gamma_F}(F, P_{\Sigma_{\gamma_F}}(\mathsf{OBS})) \in \{\text{F-présent}, \text{F-ambigu}, \text{F-absent}\}.$$

Le résultat final est un ensemble *Diag* contenant des hypothèses Δ de fautes possibles. Chaque ensemble Δ est un ensemble contenant les fautes F_i telles que $\Delta_{\gamma_F}(F_i, P_{\Sigma_{\gamma_{F_i}}}(\mathsf{OBS}))$ = F_i-présent et potentiellement des fautes F_i telles que $\Delta_{\gamma_F}(F_i, P_{\Sigma_{\gamma_{F_i}}}(\mathsf{OBS}))$ = F_i-ambigu.

Théorème 3.7 *Si $\Delta_{\gamma_F}(F)$ est correct pour toute faute F alors l'algorithme 3.8 est correct.*

6. La généralisation de cette étude aux motifs introduit des difficultés supplémentaires en cours d'étude.

Entrées : Modèle compositionnel : $\{\mathcal{A}_1, \dots, \mathcal{A}_n\}$
pour tous les F *faute* **faire**
  Sélectionner $\gamma_F \subseteq \{\mathcal{A}_1, \dots, \mathcal{A}_n\}$ et construire $\Delta_{\gamma_F}(F)$;
fin pour tous
$S \leftarrow \emptyset$;
pour tous les F *faute* **faire**
  $S \leftarrow \Delta \cup \{\Delta_{\gamma_F}(F, P_{\Sigma_{\gamma_F}}(\mathsf{OBS}))\}$;
fin pour tous
$Diag \leftarrow \{\Delta = \{F_i\}_{F_i\text{-présent} \in S} \cup \Delta_{\text{ambigu}} \ : \ \Delta_{\text{ambigu}} \subseteq \{F_i \ : \ F_i\text{-ambigu} \in S\}\}$;
retourner $Diag$

Algorithme 3.8 : Algorithme par diagnostiqueurs spécialisés

Un diagnostiqueur spécialisé est dit *correct* s'il ne se trompe pas, à savoir qu'il n'affirme pas :

— F-absent alors qu'il est possible que F ait eu lieu ;

— F-présent alors qu'il est possible que F n'ait pas eu lieu.

Il en découle qu'un diagnostiqueur spécialisé qui répond F-ambigu est toujours correct. Si le diagnostiqueur spécialisé de chaque faute F est correct, alors le théorème 3.7 garantit que l'algorithme 3.8 est correct dans le sens où l'ensemble $Diag$ est un sur-ensemble des solutions du problème de diagnostic. Tout candidat (définition 3.14) est effectivement inclus dans $Diag$.

3.4.3 Diagnostiqueur précis

La mise en œuvre d'un diagnostiqueur spécialisé à l'aide d'algorithmes comme ceux présentés en section 3.2 garantit que l'algorithme par spécialisation ne se trompe pas du point de vue logique mais il ne garantit en rien sa *précision*. La difficulté inhérente d'une telle approche est de déterminer des diagnostiqueurs spécialisés qui soient précis. Soit Γ le système des n composants $\{\mathcal{A}_1, \dots, \mathcal{A}_n\}$, et $\gamma \subseteq \Gamma$ un sous-système quelconque contenant le composant où la faute F peut se produire.

Définition 3.34 (Diagnostiqueur précis) *Un diagnostiqueur $\Delta_\gamma(F)$ est précis si pour toute séquence observable σ issue du système telle $\Delta_\Gamma(\sigma) = F$-présent, pour toute continuation observable σ' du système, il existe $n \in \mathbb{N}$ telle que $|\sigma'| \geq n \Longrightarrow \Delta_\gamma(F, P_{\Sigma_\gamma}(\sigma\sigma')) = F$-présent.*

Intuitivement, un diagnostiqueur sur un sous-système γ est précis pour une faute F s'il répond F-présent pour une séquence observable globale $\sigma\sigma'$ alors que le diagnostiqueur global sur le système Γ a répondu F-présent antérieurement. Autrement dit, le diagnostiqueur spécialisé sur γ s'accorde sur la présence de la faute avec le diagnostiqueur global de façon retardée mais ce retard est fini.

Théorème 3.8 *Si $\Delta_{\gamma_F}(F)$ est précis pour toute faute F alors l'algorithme 3.8 est précis.*

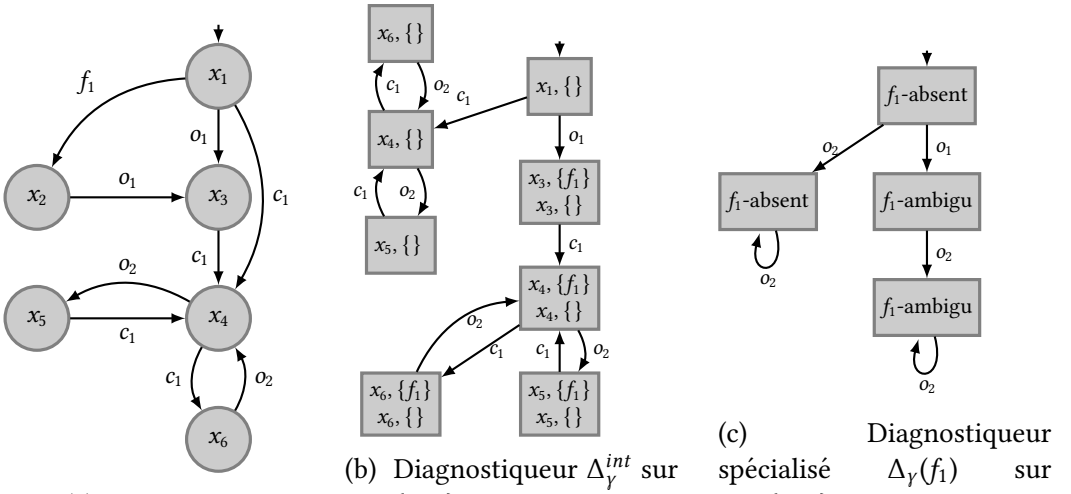
Le théorème précédent exprime l'intérêt d'un diagnostiqueur précis. En général le diagnostiqueur précis n'est déployé que sur un sous-système et non pas sur le système entier. Si n_f est le nombre de fautes anticipées dans le système, d'un point de vue complexité il est toujours préférable de gérer n_f diagnostiqueurs sur des sous-systèmes de taille moindre qu'un seul diagnostiqueur sur le système entier du fait de la complexité du problème qui est exponentielle en le nombre de composants impliqués dans le système. Par précis, le théorème 3.8 indique que le résultat de l'algorithme certifie que si la faute f a été diagnostiquée globalement comme certaine par le diagnostiqueur global alors il suffit d'attendre un nombre fini d'observations supplémentaires pour que l'algorithme assure également que la présence de la faute f est certaine. Il ne faut pas confondre ici la notion de précision et celle de diagnosticabilité. La précision est une propriété relative à la capacité du diagnostiqueur global, si le diagnostiqueur global est capable de diagnostiquer avec certitude la présence d'une faute *pour un scénario observable donné* alors le diagnostiqueur précis en sera également capable. Si le système est de plus diagnosticable alors le diagnostiqueur précis sera capable de diagnostiquer la faute avec certitude dans tous les cas (avec potentiellement un retard fini).

3.4.4 Caractérisation d'un diagnostiqueur précis

L'intérêt d'une approche par diagnostiqueurs spécialisés est de ne pas avoir à déployer les diagnostiqueurs sur tout le système mais sur une sous-partie de ce système. Si de plus, le diagnostiqueur spécialisé en question est précis sur le sous-système sur lequel il est déployé alors il est *suffisant* pour avoir autant d'informations sur le diagnostic de la faute que le diagnostiqueur global pourrait en avoir. La conséquence immédiate est que le déploiement d'un ensemble de diagnostiqueurs précis peut se faire en pratique sur des systèmes de plus grande dimension en terme de composants, systèmes sur lesquels il n'est pas possible d'exploiter un diagnostiqueur global car les ressources calculatoires sont trop limitées. Il y a tout de même un problème, à savoir, comment peut-on déterminer qu'un diagnostiqueur est précis sachant que la précision est définie par comparaison avec un diagnostiqueur global dont on impose finalement que l'on ne peut pas le calculer ?

Dans [48], on présente un critère *suffisant* pour garantir qu'un diagnostiqueur $\Delta_\gamma(F)$ est précis, sans pour autant calculer de diagnostiqueur global. Ce critère s'appuie sur l'analyse des interactions de γ avec son voisinage. Illustrons ce critère à l'aide de la figure 3.16.

À gauche, on présente un composant $\mathcal{A}_4$ qui constitue dans cet exemple le sous-système γ que l'on considère pour l'étude de la précision du diagnostiqueur spécialisé pour la faute f_1. Pour établir ce critère, on calcule le diagnostiqueur classique Δ_γ^{int} sur le



(a) Automate $\mathcal{A}_4$.

(b) Diagnostiqueur Δ_γ^{int} sur $\gamma = \{\mathcal{A}_4\}$.

(c) Diagnostiqueur spécialisé $\Delta_\gamma(f_1)$ sur $\gamma = \{\mathcal{A}_4\}$.

FIGURE 3.16 – Un composant, son diagnostiqueur interactif et son diagnostiqueur précis.

sous-système γ sous l'hypothèse que toutes les interactions du sous-système en question sont également observables. Dans le cadre de l'exemple de la figure 3.16, les observations de $\mathcal{A}_4$ sont o_1 et o_2, à cela s'ajoute l'événement interactif c_1. Au milieu de la figure 3.16, on retrouve le diagnostiqueur Δ_γ^{int} en question où l'on suppose que o_1, o_2 et c_1 sont effectivement observables. À droite de la figure 3.16, on présente enfin le diagnostiqueur spécialisé $\Delta_\gamma(f_1)$ qui a été obtenu comme suit :

1. on construit le diagnostiqueur classique Δ_γ sur γ en ne s'appuyant ici que sur les véritables observations du système à savoir o_1 et o_2 ;
2. puis pour chaque état du diagnostiqueur ainsi obtenu, on abstrait de la façon suivante l'étiquette de chaque état :
   — si aucun candidat de l'état ne contient la faute f_1, on remplace par f_1-absent ;
   — si tout candidat contient la faute f_1, on remplace par f_1-présent ;
   — sinon on remplace par f_1-ambigu.

La conclusion de cet exemple est que le diagnostiqueur spécialisé $\Delta_\gamma(f_1)$ est précis et que cette conclusion dérive *uniquement* d'une analyse comparative des deux diagnostiqueurs Δ_γ^{int} et $\Delta_\gamma(f_1)$. Considérons un chemin de transitions quelconque dans le diagnostiqueur spécialisé $\Delta_\gamma(f_1)$ (par exemple o_1 suivi de o_2). Ce chemin est associé à plusieurs chemins dans le diagnostiqueur Δ_γ^{int} qui observe les interactions, il s'agit des chemins :

1. $\xrightarrow{o1}\xrightarrow{c_1}\xrightarrow{o_2}$ qui mène à l'état $(x_5, \{f_1\}), (x_5, \{\})$;
2. $\xrightarrow{o1}\xrightarrow{c_1}\xrightarrow{c_1}\xrightarrow{o_2}$ qui mène à l'état $(x_4, \{f_1\}), (x_4, \{\})$.

Les deux chemins mènent à un état qui est f_1-ambigu, autrement dit, dès lors que l'on observe effectivement $o_1 o_2$, les occurrences de c_1 (que cela soit une occurrence ou deux occurrences) n'affectent en rien le résultat, l'interaction c_1 ne *discrimine* pas dans ce scénario vis-à-vis du diagnostic de f_1. Si maintenant on considère un autre chemin du diagnostiqueur spécialisé de droite (par exemple o_2 suivi de o_2), là encore les chemins du diagnostiqueur Δ_γ^{int} produisant o_2 suivi de o_2 mènent tous vers des états f_1-absents. Dans les deux cas, les chemins de Δ_γ^{int} associés mènent tous vers des états dits f_1-*équivalents* (soit ils sont tous f_1-ambigus, soit f_1-présents, soit f_1-absents). C'est exactement le critère qui assure la précision du diagnostiqueur.

Théorème 3.9 *Le diagnostiqueur $\Delta_\gamma(f_1)$ est précis, si pour tout chemin $\tau = \xrightarrow{o_1} \cdots \xrightarrow{o_n}$ de $\Delta_\gamma(f)$, les chemins $\tau' = \xrightarrow{e_1} \cdots \xrightarrow{e_m}\xrightarrow{o_n}$ de Δ_γ^{int} tels que $P_{\Sigma_\gamma \to \Sigma_o}(e_1 \dots e_m o_n) = o_1 \dots o_n$ mènent à des états f-équivalents.*

Il est à noter que ce critère n'est que suffisant. Il existe en effet des diagnostiqueurs spécialisés et précis qui ne vérifient pas ce critère. Pour s'en convaincre, il suffit d'imaginer que les chemins du diagnostiqueur Δ_γ^{int} ne respectant pas ce critère ne soient pas globalement cohérents (ils ne se réalisent pas globalement car non synchronisables). Ne se réalisant pas, ils n'induisent aucun problème d'imprécision bien que le critère ne soit pas respecté. De ce critère, on peut néanmoins obtenir le résultat général suivant.

Théorème 3.10 ([56]) *Si les interactions de γ avec son voisinage sont toutes observables alors $\Delta_\gamma(f_1)$ est précis.*

Il est en effet inutile d'exploiter dans ce cas les observations des autres composants. L'observation de ces autres composants a en général pour objectif de confirmer ou d'infirmer de potentielles synchronisations avec γ mais comme dans ce cas spécifique, toute synchronisation de γ avec ces composants est observable, c'est devenu inutile. C'est grâce à cette propriété qu'il a notamment été possible de définir un diagnostiqueur modulaire à base de réseaux de Petri colorés qui soit précis pour chaque module dans [47].

3.4.5 Diagnostiqueur précis et abstraction de connexions

La section précédente présente un moyen de déterminer si un diagnostiqueur est précis sur un sous-système donné. Elle a mis en avant le fait que la précision d'un tel diagnostiqueur s'appuyait sur une analyse des interactions entre le sous-système et son environnement. Une interaction par un événement e entre composants telle qu'elle a été définie en section 3.2.2.1 représente en fait une connexion $c_e = \langle e_1, e_2 \rangle$ entre ces deux composants où e_1, l'événement du premier composant, est systématiquement synchronisé avec l'événement e_2 du deuxième composant (événement synchronisé que l'on note en général $e = e_1 \| e_2$). Dans [28] et

[29], on bénéficie de la relation entre précision et interaction pour améliorer encore plus l'efficacité des diagnostiqueurs spécialisés en analysant les connexions internes au sous-système qui n'apportent rien au niveau de la précision. Si une connexion n'apporte rien, il est inutile de l'exploiter, on peut donc l'abstraire et ainsi réduire les temps de calcul [Sachenbacher and Struss, 2005]. Ceci est notamment plus intéressant dès lors que l'abstraction d'une connexion casse un cycle de synchronisation (un anneau de composants connectés) ce qui permet d'optimiser les synchronisations entre composants en utilisant une stratégie de propagation d'arbres de jonction ([Jensen and Jensen, 1994], [Huang and Darwiche, 1996], [Kan John and Grastien, 2008], [Schumann and Huang, 2008], [28]). Un sous-système avec des connexions inactives est une *configuration*.

Définition 3.35 (Configuration) *Une configuration est un triplet* $\mathbb{C} = (\gamma, C, \overline{C})$ *où* γ *est un sous-système,* C *est l'ensemble des connexions internes/externes actives de* γ *et* $\overline{C}$ *sont les connexions inactives.*

Dans une configuration, chaque composant a potentiellement des synchronisations inactives, donc l'ensemble de ses comportements admissibles augmentent. Soit l'ordre $\prec_\gamma$ tel que $\mathbb{C} \prec_\gamma \mathbb{C}'$ si $\mathbb{C} = (\gamma, C, \overline{C})$ et $\mathbb{C}' = (\gamma, C', \overline{C}')$ et $C \subset C'$, $C \cup \overline{C} = C' \cup \overline{C}'$.

Théorème 3.11 *Soit deux configurations* $\mathbb{C}$ *et* $\mathbb{C}'$ *telles que* $\mathbb{C} \prec_\gamma \mathbb{C}'$*, pour tout composant de* γ*, l'ensemble des trajectoires admissibles du composant dans la configuration* $\mathbb{C}$ *contient l'ensemble des trajectoires admissibles du même composant dans la configuration* $\mathbb{C}'$*.*

Ainsi, la désactivation d'une connexion augmente l'ensemble des comportements admissibles du modèle (mais pas du système où la connexion est bien évidemment toujours présente). La question qui reste à élucider est de savoir si la désactivation de la connexion a des conséquences sur la précision du diagnostiqueur. On ne définit plus un diagnostiqueur par sous-système mais par configuration. Déterminer si un diagnostiqueur est précis pour une configuration donnée s'appuie sur le critère défini dans la section précédente.

3.5 Résumé

Le problème du diagnostic de systèmes à événements discrets est un problème générique avec de multiples applications qui offre un champ d'investigation et de recherche riche et varié. La difficulté principale dans la résolution de ce type de problème est l'explosion combinatoire liée au nombre de composants du système à traiter. Une première contribution a consisté à mettre en place des algorithmes symboliques de diagnostic mettant à profit l'utilisation de diagrammes de décision binaires afin de factoriser les trajectoires et ainsi augmenter l'efficacité des algorithmes en pratique. Le

deuxième point traité concerne la diagnosticabilité des systèmes. Cette propriété importante permet de caractériser l'utilité des algorithmes de diagnostic précédemment définis. En effet, si l'on sait qu'un système est peu diagnosticable, il n'y a guère d'intérêt d'y déployer des algorithmes de diagnostic aussi efficaces soient-ils, étant donnés qu'ils répondront toujours avec ambiguïté. Les contributions proposées ici pour cette analyse prennent en compte l'explosion combinatoire sous-jacente de différentes manières : soit l'analyse est menée de façon décentralisée afin d'éviter l'analyse globale, soit elle met à profit des outils de vérifications de modèle gérant l'explosion combinatoire par des techniques d'ordres partiels et d'ensemble persistants telles que l'outil TINA les propose. C'est d'ailleurs à l'aide de cet outil que le problème de diagnosticabilité a été résolu pour des comportements plus complexes que de simples fautes. Un point important dans l'analyse de diagnosticabilité est qu'une réponse binaire ne peut être une fin en soi et que en cas de non-diagnosticabilité, il est essentiel d'apporter des éléments de solutions pour améliorer cette diagnosticabilité. Enfin, parmi ces contributions, nous avons introduit le concept de précision et de diagnostic par spécialisation. Là encore, on part du constat que le diagnostic d'une faute ne peut s'effectuer qu'en ne s'appuyant que sur une sous-partie du système. L'analyse de précision a cet objectif : que le système soit diagnosticable ou pas, un diagnostiqueur précis certifie que son résultat sera identique à celui obtenu globalement. Cette analyse conduit également à fournir de nouveaux moyens d'abstractions de modèles telle que l'inactivation de connections entre composants.

Chapitre 4

Diagnostic temporel

L'une des quantités physiques les plus faciles à mesurer est le temps : le capteur type pour le mesurer est l'*horloge*. Cette facilité de mesure doit être mise à profit dans un raisonnement de diagnostic. Cette dimension vient enrichir la connaissance qui a été exploitée dans le chapitre 3. Dans les systèmes dynamiques qui ont été étudiés jusqu'à présent, le temps était caractérisé par la séquence d'événements (un événement b se produit *après* un événement a). Il n'est nullement fait mention de la dimension quantitative du temps, des notions de durées et de dates. Le temps a une importance considérable dans un raisonnement diagnostique car le temps peut être symptomatique d'un problème dans le système. Par exemple, si un système, tel qu'un serveur web, est supposé répondre à une requête r par l'envoi d'une page web p, peut-on dire que les deux scénarios observables suivants sont symptomatiquement équivalents ?

1. envoi de la requête r à la date d, réception de p à la date d + 1ms ;

2. envoi de la requête r à la date d, réception de p à la date d + 1 an.

Dans les deux cas, on observe bien r suivi de p avec p qui arrive en un temps fini, ainsi du point de vue purement séquentiel, les deux scénarios observables sont totalement identiques. Par contre, en considérant l'information quantitative du temps comme une observation, il devient évident que le deuxième scénario est anormal, seule la mesure du temps discrimine dans ce cas. C'est tout l'objet de ce chapitre où sont décrites les différentes contributions liées à l'intégration du temps dans le raisonnement diagnostique sur les systèmes dynamiques.

4.1 Diagnostic temporel et reconnaissance de chroniques

4.1.1 Contexte des travaux

La présente section a pour objectif de présenter le problème de diagnostic temporel dans son ensemble, il s'agit donc d'une introduction et d'un état de l'art servant à présenter les contributions à ce problème. Néanmoins, la façon de décrire le problème en soi est originale car elle est plus générique que celle présente dans la littérature antérieure. La présentation des systèmes à événements discrets temporels ainsi qu'une partie de la description du problème sont issues des travaux de thèse de Houssam-Eddine Gougam.

4.1.2 Systèmes à événements discrets temporels

Nous reprenons ici les éléments qui ont été introduits dans la section 3.1.1, éléments auxquels l'information temporelle est ajoutée.

Définition 4.1 (Événement daté) *Un événement daté est un événement e auquel on adjoint sa date d'occurrence $d \in \mathbb{R}^+$, on le note (e, d).*

Un système à événements discrets temporel produit des comportements constitués d'événements datés. L'événement daté est ainsi constitué de son *type* $e \in \Sigma$ où Σ est l'ensemble de types d'événements comme dans les SED classiques et de sa date d'occurrence d.

Définition 4.2 (Point temporel) *Un point temporel est une date particulière $d \in \mathbb{R}^+$, on le note (λ, d).*

L'événement λ est un événement fictif qui ne correspond pas à un événement physique propre au système, il permet juste d'exprimer un point dans le temps, une date.

Définition 4.3 (Comportement temporel) *Un comportement temporel est une séquence d'événement datés de $\Sigma \cup \{\lambda\}$.*

L'ensemble des séquences temporelles sur Σ est noté $\mathcal{T}(\Sigma)$ et

$$\mathcal{T}(\Sigma) \subseteq (\mathbb{R}^+ \times \Sigma \cup \{\lambda\})^+.$$

On peut noter une séquence temporelle soit en date absolue (la date d de l'événement daté (e, d) est relative à l'origine des temps), soit en date relative (la date d de l'événement daté (e, d) est relative à la date de l'événement qui le précède). Par exemple, si l'origine des temps est 0, les deux notations suivantes sont équivalentes :

— en date absolue : $(a, 1).(b, 3).(\lambda, 5).(a, 8).(\lambda, 9)$;
— en date relative : $(a, 1).(b, 2).(\lambda, 2).(a, 3).(\lambda, 1)$.

Par la suite, les séquences temporelles seront notées en date relative (sauf mention contraire explicite) et seront notées de façon simplifiée. Par exemple, la séquence $(a, 1).(b, 2).(\lambda, 2).(a, 3).(\lambda, 1)$ pourra être notée $1a2b2\lambda3a1\lambda$. Enfin, la notation λ est utilisée seulement pour une explicitation du temps, si un tel événement $d_1\lambda$ est suivi d'un autre événement d_2e dans une séquence donnée, la séquence en question peut être réécrite de façon équivalente en remplaçant $d_1\lambda d_2e$ par $(d_1 + d_2)e$. Ainsi la séquence $1a2b2\lambda3a1\lambda$ peut se réécrire $1a2b5a1\lambda$. Finalement, on peut remarquer que tout mot de $\mathcal{T}(\Sigma)$ contient au moins un couple (e, d). La notion de séquence vide ε propre au système atemporel s'exprime ici de façon différente : le comportement exprimant le seul démarrage du système est noté 0λ, le temps peut courir sans que le système ne produise un événement ce qui s'exprime par $d\lambda$ pour une date d donnée $d > 0$. Chaque séquence temporelle a une *forme canonique* à savoir $d_1e_1.d_2e_2 \ldots d_ne_nd_{n+1}\lambda$ où les e_i sont des événements de Σ. On note durée(τ) la durée totale de la séquence temporelle τ, par exemple durée$(1a2b5a1\lambda) = 9$, si τ est sous forme canonique durée$(\tau) = \Sigma_{i=1}^{n+1} d_i$.

Un SED temporel est caractérisé par un ensemble de séquences temporelles.

Définition 4.4 (Modèle du système SED temporel) *Un système à événements discrets temporel est représenté par son langage de comportements $\mathcal{S} \subseteq \mathcal{T}(\Sigma)$.*

Là encore, comme dans le chapitre 3, nous ne nous limitons pas au choix d'un formalisme pour représenter ce type de système. Ceci est d'autant plus vrai que dans le cadre temporel, chaque formalisme va imposer des restrictions d'expressivité sur la nature du système modélisé liées à des problèmes de décidabilité.

4.1.3 Problème de diagnostic temporel

Le problème de diagnostic qui est présenté ici intègre la dimension temporelle au problème défini dans la section 3.1.1. Il faut donc revisiter les notions d'observations et y intégrer l'observation du temps. Les phénomènes que l'on cherche à expliquer peuvent également être de nature temporelle.

La notion de masque observable événementiel $obs : \Sigma \longrightarrow 2^{\Sigma_o \cup \{\varepsilon\}}$ reste inchangée (voir la définition 3.6) car elle associe aux événements générés par le système des événements observés, par contre la notion de séquence observable change et prend maintenant en compte la dimension temporelle. Une séquence observable appartient désormais à $\mathcal{T}(\Sigma_o)$. Soit τ une trace du système $\tau \in \mathcal{S} \subseteq \mathcal{T}(\Sigma)$ sous sa forme canonique, on définit $obsT(\tau, obs) \subseteq \mathcal{T}(\Sigma_o)$ comme la projection temporelle du masque observable événementiel de τ. $obsT$ est le *masque observable temporel* du système, il est défini comme suit.

1. $obsT(\delta\lambda, obs) = \{\delta\lambda\}$, pour tout $\delta \in \mathbb{R}^+$;

2. si $\tau = \delta e.\tau_1$ et $\tau_1 \in \mathcal{T}(\Sigma)$ alors $obsT(\delta e.\tau_1, obs) = obsT_1 \cup obsT_2$ avec

$$obsT_1 = \bigcup_{o \in obs(e), o \neq \varepsilon} \delta o.obsT(\tau_1);$$

et
— si $\varepsilon \notin obs(e)$, $obsT_2 = \emptyset$.
— si $\varepsilon \in obs(e)$,

$$obsT_2 = \{(\delta + \delta')o'.\tau' \text{ si } \delta' o'.\tau' \in obsT(\tau_1, obs)\}$$

$$\cup\{(\delta + \delta')\lambda \text{ si } \delta'\lambda \in obsT(\tau_1, obs)\}.$$

Tous les éléments présentés ci-dessus, associés ensemble, définissent l'extension temporelle du *modèle observé*.

Définition 4.5 (Modèle du système temporel observé) *Le modèle d'un système observé est un triplet* $\mathsf{DS} = (\mathcal{S}, \Sigma_o, obsT)$ *tel que :*
— $\mathcal{S}$ *est le modèle d'un système temporel sur un alphabet* Σ *;*
— Σ_o *est un ensemble d'événements observables ;*
— $obsT : \mathcal{T}(\Sigma) \times 2^{\Sigma_o \cup \{\varepsilon\}} \longrightarrow \mathcal{T}(\Sigma_o)$ *est le masque observable temporel.*

Les phénomènes que l'on cherche à expliquer peuvent également être de nature temporelle, ce qui se traduit dans la notion de motifs temporels.

Définition 4.6 (Motif temporel) *Un motif temporel* M *sur un alphabet* Σ_M *est un langage sous-ensemble de* $\mathcal{T}(\Sigma_M)$ *tel que si* $\tau \in M$ *alors toute continuation de* τ *n'est pas dans ce langage.*

Le principe d'un motif temporel est de représenter des comportements temporels constitués d'événements de Σ_M. Par continuation, on entend que si τ est un mot du motif, il n'existe pas de mot $\tau' \in \mathcal{T}(\Sigma_M)$ tel que $\tau.\tau'$ soit également un mot de ce motif : cette propriété représente le fait que dès que le motif est reconnu, il le sera toujours. À titre d'exemple, la simple occurrence d'une faute f est un motif temporel, le langage de ce motif est $\{df : d \geq 0\}$.

Comme dans le cadre atemporel, un état de santé Δ est un ensemble de motifs (voir la définition 3.10) qui *ont eu lieu* sur le système, ce qui est formalisé par la notion de concordance temporelle.

Définition 4.7 (μ-concordance temporelle) *Soit* τ *et* μ *deux séquences temporelles sur un alphabet* Σ *avec* μ *sous forme canonique, la séquence* τ *est* μ*-concordante si*

1. $\mu = d\lambda$ *et* durée$(\tau) \geq d$ *;*
2. $\mu = d_0 e_0 \mu_1$, $d_0 \in \mathbb{R}^+$, $e_0 \in \Sigma$ *et il existe deux séquences temporelles* τ_0, τ_1 *et une durée* $d \in \mathbb{R}^+$ *telles que :*

(a) $\tau = \tau_0 de_0 \tau_1$ *;*

(b) $d_0 = \text{durée}(\tau_0) + d$ *;*

(c) τ_1 *est* μ_1*-concordante.*

Le problème général du diagnostic dans les SED temporels peut maintenant être posé.

Définition 4.8 (Problème de diagnostic de SED temporel) *Un problème de diagnostic temporel est un triplet* $(\mathsf{DS}, \mathcal{M}, \mathsf{OBS})$ *:*

- DS *est le modèle d'un système temporel observé sur un alphabet* Σ *et sur un alphabet observable* Σ_o *;*
- $\mathcal{M}$ *est un ensemble de motifs temporels d'intérêt sur un alphabet* $\Sigma_{\mathcal{M}} \subseteq \Sigma$ *;*
- OBS *est une séquence temporelle d'observations (*$\mathsf{OBS} \in \mathcal{T}(\Sigma_o)$*).*

La solution à ce problème est l'ensemble des diagnostics candidats.

Définition 4.9 (Diagnostic candidat) *Un diagnostic candidat de* $(\mathsf{DS}, \mathcal{M}, \mathsf{OBS})$ *est un état de santé* Δ *s'il existe une trajectoire* $\tau \in \mathcal{S}$ *telle que* $\text{durée}(\tau) = \text{durée}(\mathsf{OBS})$ *et*

1. $\mathsf{OBS} \in obsT(\tau, obs)$ *; et*
2. $M \in \Delta$ *ssi* τ *est* M*-concordante.*

Dans le chapitre précédent, d'autres formes de diagnostics candidats ont été définis, prenant en compte notamment une estimation d'états. En temporel, il est tout à fait possible de définir ces problèmes mais leur résolution pose des problèmes de décidabilité dûs à la continuité du temps et donc à l'infinité du nombre d'états du SED temporel sous-jacent. L'estimation d'états ne peut être énumérative comme dans le cas atemporel, des abstractions d'états sont toujours nécessaires.

4.1.4 Reconnaissance de chroniques

Dans le monde du diagnostic de systèmes dynamiques temporels, il existe majoritairement deux grandes familles de techniques. La première d'entre elle consiste à modéliser le comportement du système avec un formalisme adéquat (automate temporel, réseau de Petri temporel). Cette technique, contrairement au SED classique, a des développements plus récents. Les travaux peuvent s'appuyer sur des automates temporisés [Tripakis, 2002, Rozé and Cordier, 2002, Bouyer et al., 2005, Khoumsi and Ouedraogo, 2009] sur des réseaux de Petri où le temps est représenté dans les places [Bonhomme, 2015] ou sur les transitions [Ghazel et al., 1999, Chatain and Jard, 2005, Jiroveanu and Boel, 2006, Boel and Jiroveanu, 2013, Basile et al., 2014, Liu et al., 2014a, Wang et al., 2015]. La deuxième grande famille de techniques qui ont été développées au fil des années

approche le diagnostic temporel d'une façon symptomatique et ne s'appuie pas sur la caractérisation formelle complète du système. Un symptôme est un ensemble d'événements observés auquel on associe la présence certaine d'un dysfonctionnement, d'une faute. Le principe clé de ces techniques est que le processus de diagnostic consiste à reconnaître dans le flux d'observations datées généré par le système, la présence du symptôme. Si la présence du symptôme est effective (on dit également que le symptôme est reconnu), alors le processus de diagnostic conclut que la faute associée au symptôme reconnu est un diagnostic candidat. Parmi les formalismes utilisés pour modéliser ces symptômes, on peut citer notamment les *chroniques* [Dousson et al., 1993, Dousson and Duong, 1999b, Guerraz and Dousson, 2004] et les *signatures temporelles causales* [Toguyéni et al., 1990, Saddem and Philippot, 2014], les motifs de conditions [Pandalai and Holloway, 2000]. Les contributions apportées au domaine s'appuient sur le formalisme des *chroniques*, ce formalisme est utilisé dans de nombreux domaines, notamment dans la supervision de réseaux de télécommunication [Dousson and Duong, 1999a, Dousson and Duong, 1999b, Cordier and Dousson, 2000], de réseaux de distribution électrique [Laborie and Krivine, 1997b] ou de services Web [Le Guillou et al., 2008, Vizcarrondo et al., 2013] [18], la reconnaissance de situations en planification [Dousson et al., 1993, Ghallab, 1996], la détection d'intrusion [Morin and Debar, 2003], la simulation d'architectures abstraites [Choppy et al., 2009], ou encore dans l'analyse d'électrocardiogrammes pour l'analyse d'arythmies cardiaques [Carrault et al., 1999].

Le formalisme des chroniques offre un moyen de représenter l'occurrence d'événements temporellement contraints par des intervalles [Allen, 1983, Allen, 1984], outil qui est utilisé pour raisonner sur les actions ou sur des changements de situations [Kowalski and Sergot, 1986]. Il fait partie des systèmes de reconnaissances de situations comme ceux de [Nokel, 1989, Haimowitz and Kohane, 1993, Kockskämper et al., 1994, Gamper and Nejdl, 1997]. Dans [Dousson et al., 1993], une chronique est représentée sous forme de prédicats, plus précisément :

1. un ensemble de prédicats ;
2. un ensemble de contraintes temporelles qui concernent ces prédicats ;
3. un ensemble d'actions à faire lorsque la chronique est reconnue, cette dernière partie étant optionnelle.

Pour être reconnue, une chronique doit satisfaire tous les prédicats et les contraintes qu'elle contient. Les prédicats sont de trois types.

1. `event` : ce prédicat correspond à un changement de la valeur d'un attribut ou une occurrence d'un certain message. Par exemple,

$$\texttt{event}(\texttt{M}, \texttt{t}), \texttt{event}(\texttt{A} : (\texttt{a}_1, \texttt{a}_2), \texttt{t}),$$

le premier prédicat exprime l'occurrence de l'événement M à l'instant t et le deuxième prédicat exprime l'occurrence d'un événement de type *changement de la valeur de l'attribut A de a_1 à a_2* au temps t.

2. `noevent` : ce prédicat exprime une contrainte sur la non-occurrence d'un événement d'un type donné entre deux dates. Par exemple,

$$\texttt{noevent}(\texttt{A} : (\texttt{a}_1, \texttt{a}_2), (\texttt{t}_1, \texttt{t}_2)), \texttt{noevent}(\texttt{M}, (\texttt{t}_1, \texttt{t}_2)),$$

le premier prédicat interdit le changement de la valeur de l'attribut A de a_1 à a_2 dans l'intervalle de temps $[t_1, t_2]$ et le second interdit l'apparition d'un message M dans l'intervalle de temps $[t_1, t_2]$.

3. `occurs` : ce prédicat a été introduit plus récemment dans [Dousson, 2002]. C'est un prédicat de comptage. Par exemple,

$$\texttt{occurs}((\texttt{n}, \texttt{m}), \texttt{a}, (\texttt{t}_1, \texttt{t}_2)),$$

ce prédicat représente l'occurrence d'un événement a, n à m fois entre les instants t_1 et t_2. C'est le plus puissant des prédicats disponibles, puisqu'il peut remplacer les deux précédents. Il permet d'unifier la représentation des chroniques sous un seul prédicat (le $\texttt{noevent}(a, t_1, t_2)$ pouvant être représenté par $\texttt{occurs}((0, 0), \texttt{a}, (\texttt{t}_1, \texttt{t}_2))$).

L'intérêt des chroniques, outre le fait qu'elles représentent des comportements temporels, est l'existence d'outils qui sont en mesure d'analyser des séquences temporelles observables et de déterminer si les chroniques sont effectivement reconnues [Dousson et al., 1993, Choppy et al., 2009]. Le deuxième intérêt des chroniques est qu'elles offrent une représentation du système de surface. Elles ne s'appuient en effet sur aucun modèle comportemental formel, elles ne représentent que des motifs observables. Les chroniques ont souvent été modélisées par expertise dans un premier temps mais il existe désormais des travaux sur leur synthèse automatique par extraction/fouille [Mannila et al., 1997, Dousson and Duong, 1999a, Fessant et al., 2004, Mitsa, 2010, Guyet and Quiniou, 2011], par abstraction [Guerraz and Dousson, 2004] ou encore par apprentissage automatique [Mayer, 1998, Cram et al., 2011, Subias et al., 2014, Vasquez et al., 2017].

L'un des outils de reconnaissance s'appelle CRS (*Chronicle Recognition System*) [Dousson et al., 1993]. Le principe de la reconnaissance de chroniques est illustré sur la figure 4.1.

En haut à gauche de la figure, la forme effective d'une chronique utilisée dans CRS est présentée, une forme graphique de cette même chronique est en bas à gauche. À droite, une séquence temporelle d'observations est présentée. Sur cette séquence, on y présente deux *instances de chroniques*, la première est *partielle*, seuls deux des trois événements attendus remplissent les contraintes, le troisième événement arrivant trop

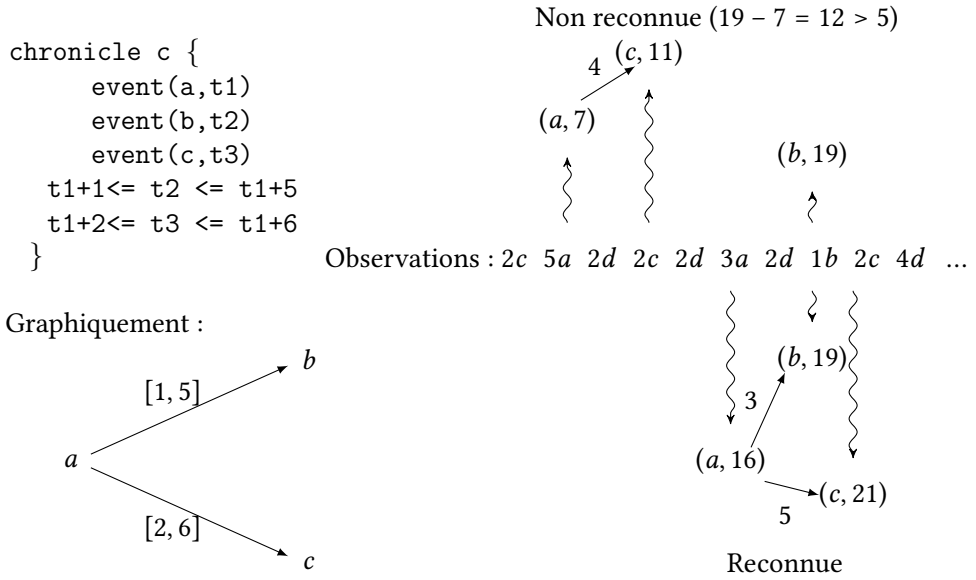


FIGURE 4.1 – Principe de la reconnaissance de chroniques.

tard : cette instance partielle ne donne pas lieu à la reconnaissance de cette chronique. La deuxième instance est, quant à elle, *complète* et permet de conclure à la reconnaissance de la chronique. Le principe de reconnaissance présenté ici est centralisé (un seul reconnaisseur) mais il est tout à fait possible de distribuer la reconnaissance [Boufaied et al., 2004, Le Guillou et al., 2008, Vizcarrondo et al., 2013].

4.2 Modélisations et analyses de chroniques

4.2.1 Contexte de ces travaux

La modélisation étendue des chroniques ainsi que l'analyse de la qualité des chroniques a été proposée dans le cadre des travaux de stage de Master 2 de Ghyslain Maître. Le travail sur les chroniques exclusives et leur lien avec la diagnosticabilité du système a été initié au cours d'une collaboration avec Audine Subias afin de répondre à un projet européen, le projet WS-DIAMOND. Ce travail a ensuite été développé, toujours en collaboration avec Audine Subias, dans le cadre de l'encadrement des stages de Master 2 de Houssam-Eddine Gougam et de Ghyslain Maître.

4.2.2 Formalisation des chroniques et de leur reconnaissance pour le diagnostic

La notion de chronique a été introduite et développée dans [Dousson et al., 1993, Ghallab, 1996, Dousson and Duong, 1999b, Dousson, 2002] qui a réalisé le moteur de reconnaissance CRS. La section 4.1.4 a présenté un langage à base de prédicats pour modéliser une telle chronique (aspect syntaxique) mais ne présente pas formellement la sémantique d'une chronique et la sémantique de sa reconnaissance. Dans les travaux de [Dousson et al., 1993], la sémantique de la reconnaissance est liée à l'outil, pour savoir si une chronique est reconnue sur un flux, il faut utiliser CRS qui répondra oui ou non. Afin de faire des analyses plus poussées sur les chroniques et leur capacité de diagnostic, il faut modéliser formellement la *reconnaissance de chroniques* autrement dit, il faut utiliser un langage formel qui puisse exprimer comment CRS va effectivement reconnaître une chronique sur un flux donné. Un deuxième point important concerne l'expressivité des chroniques. La définition formelle initiale reposait uniquement sur le prédicat `event`, ainsi une chronique ne définissait que des contraintes *positives* (présence d'un événement). Dès lors que les chroniques ont été utilisées pour le diagnostic, il est vite apparu évident que ce prédicat seul ne pouvait suffire, il fallait pouvoir définir au sein d'une chronique des contraintes *négatives* (absence d'événement), d'où l'introduction dans CRS du prédicat `noevent` pour lequel la littérature ne propose pas de caractérisation formelle.

Dans [31], l'objectif était ainsi de proposer une *définition formelle* des chroniques qui reprend le cadre initial de [Dousson et al., 1993] auquel est ajoutée la notion de *contrainte d'interdiction* (notion représentée dans le langage des chroniques par le prédicat `noevent`). Une chronique est ainsi définie par un ensemble de types d'événements associé à des variables temporelles, un ensemble de contraintes temporelles entre ces variables et des contraintes d'interdiction.

Définition 4.10 (Chronique) *Une chronique est un 6-uplet,* $\mathcal{C} = (\mathcal{X}, \mathcal{A}, \mathcal{T}, \mathcal{E}, \mathcal{M}, \mathcal{F})$*, où :*

- $\mathcal{X}$*, un ensemble fini de variables temporelles ;*
- $\mathcal{A} \subseteq \mathcal{X} \times \mathcal{X}$*, un ensemble fini d'arcs ;*
- $\mathcal{T} : \mathcal{A} \longrightarrow \mathbb{I}$*, l'application qui à chaque arc associe un intervalle temporel. Cet intervalle représente la contrainte entre deux variables temporelles. Pour plus de simplicité, on notera* $\mathcal{T}(x_i, x_j) = \mathcal{T}_{ij}$ *;*
- $\mathcal{E}$*, un ensemble de types d'événements utilisé dans la chronique ;*
- $\mathcal{M} : \mathcal{X} \longrightarrow \mathcal{E}$*, une fonction de typage qui associe à chaque variable temporelle de* $\mathcal{X}$ *un type d'événement de* $\mathcal{E}$ *;*
- $\mathcal{F} : \mathcal{E} \longrightarrow 2^{\{\mathcal{X}\times\mathcal{X}\times\mathbb{Q}^2\}}$*, les contraintes d'interdiction sur les types d'événements de* $\mathcal{E}$*. Une contrainte* $(x_i, x_j, \alpha, \beta) \in \mathcal{F}(v)$*, signifie qu'on interdit* v *sur l'intervalle* $J = [x_i + \alpha, x_j + \beta]$ *où* $\alpha, \beta \in \mathbb{Q}$*.*

Cette définition étend la définition de [Dousson, 2002]. Le triplet sous-jacent $(\mathcal{X}, \mathcal{A}, \mathcal{T})$ est un problème temporel simple (STP) au sens de [Dechter et al., 1991]. Un STP est un ensemble fini de variables temporelles $\mathcal{X} = \{x_1, \dots x_n\}$ ayant des domaines continus et un ensemble fini d'intervalles $\mathcal{T}$ représentant les contraintes temporelles entre ces variables : chaque intervalle $T_{ij} = [a_{ij}, b_{ij}] \in \mathcal{T}$, $a_{ij}, b_{ij} \in \mathbb{Q}$ représente la contrainte sur la valeur permissive pour la distance $x_j - x_i$, à savoir $x_j - x_i \leq b_{ij}$ et $x_i - x_j \leq -a_{ij}$. Pour obtenir une chronique, on ajoute au STP initial trois autres types d'information. Une chronique est associée à un ensemble de type d'événements $\mathcal{E}$, événements qui émanent du masque observable du système (ou de la classe de systèmes). Chaque variable temporelle de $\mathcal{X}$ est ainsi associée à un type d'événement de $\mathcal{E}$ à l'aide de la fonction de typage $\mathcal{M}$. Enfin, on y adjoint les contraintes d'interdiction $\mathcal{F}$. Une contrainte d'interdiction représente l'absence d'un type d'événement v pendant un ou plusieurs intervalles de temps. Les bornes de ces intervalles auront pour origine les variables temporelles d'un graphe de contraintes temporelles. Soient deux variables temporelles x_i et x_j, ces intervalles seront de la forme $J = [x_i + \alpha, x_j + \beta]$ représentant respectivement le début de l'interdiction et la fin de l'interdiction avec $\alpha, \beta \in \mathbb{Q}$ et $x_i + \alpha < x_j + \beta$. Si $x_i + \alpha \geq x_j + \beta$, alors la contrainte n'a aucun effet. Soit l'instanciation $\{x_i = t_i, x_j = t_j\}$ solution du graphe et δ_v l'ensemble des dates du type d'événement v, une contrainte est respectée si $\delta_{v_k} \notin [t_i + \alpha, t_j + \beta], \forall k$. Une chronique est dite positive si elle ne contient pas de contraintes d'interdiction.

La figure 4.2 présente une chronique au sens de la définition 4.10. Cette chronique représente, à titre illustratif, un comportement dans une voiture : allumage du moteur (on), suivi, entre 10 et 20 minutes plus tard de l'activation du clignotant droit (cl_dr), et, dans un délai de 15 à 30 minutes, de l'activation du clignotant gauche (cl_ga), arrêt du moteur (off) entre 5 et 25 minutes après une activation du clignotant gauche, et, entre 5 minutes après l'allumage du moteur et 2 minutes avant l'activation du clignotant droit, l'activation du clignotant gauche est interdite. Les contraintes d'interdiction sont représentées par une transition en pointillés labellisée par le type d'événement, et les valeurs α et β. Les transitions en traits pleins sont étiquetées par les contraintes temporelles.

Spécifions maintenant formellement la reconnaissance d'une chronique. Dans un problème temporel simple constitué de n variables temporelles $\mathcal{X} = (x_1, \dots, x_n)$, le n-uplet $T = (t_1, \dots, t_n)$ est appelé une *solution* du STP, si l'instanciation $\{x_1 = t_1 \dots x_n = t_n\}$ satisfait toutes les contraintes. Dans une chronique, on adjoint à ces variables temporelles un type d'événements $\mathcal{M}(x_i) = e_i$, ainsi une solution pour une chronique positive est une séquence temporelle : $(e_1, t_1) \dots (e_n, t_n) \in \mathcal{T}(\mathcal{E})$ (notation ici en date absolue). Si l'on ajoute à cette chronique positive des contraintes d'interdiction, on restreint l'ensemble des solutions à celles qui respectent l'ensemble de contraintes. En résumé, une chronique représente implicitement un ensemble de séquences temporelles.

Proposition 4.1 *Une chronique C sur un ensemble de types d'événements $\mathcal{E}$ est un motif*

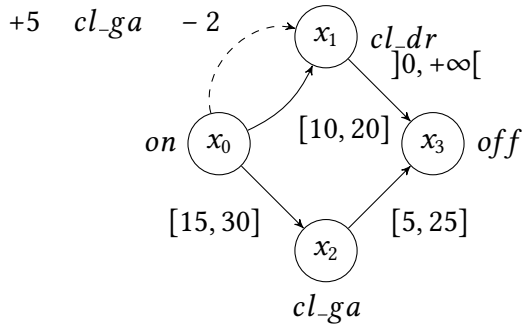


FIGURE 4.2 – Chronique C_1

temporel sur $\mathcal{E}$.

Dans le cadre d'un système de diagnostic par reconnaissance de chroniques, on peut voir ainsi une chronique comme un motif temporel particulier où tous les événements du motif font partie du masque observable du système, ce qui nous permet de déterminer un premier lien entre la reconnaissance de chroniques et le problème du diagnostic temporel.

Proposition 4.2 (Reconnaissance d'une chronique sur un flux observable) *Soit* C *une chronique, la chronique est reconnue sur le flux d'observation* $\sigma \in \mathcal{T}(\Sigma_o)$ *si et seulement si* σ *est* C*-concordante.*

Ainsi, du fait de la C-concordance, il est possible d'associer toute chronique à un ensemble de séquences temporelles observables qui peuvent être générées par le système sous-jacent. Mais chaque séquence temporelle observable est elle-même associée à un ensemble de séquences temporelles du système à travers le masque observable du système.

Proposition 4.3 (Reconnaissance d'une chronique sur un système) *Une chronique* C *est reconnaissable sur un système, s'il existe une trajectoire* $\tau \in S$ *du système telle que* $\mathsf{OBS} \in obsT^*(\tau)$ *est* C*-concordante.*

Ce dernier résultat caractérise formellement le lien entre la reconnaissance de chroniques et un problème de diagnostic temporel. Toute chronique C est associée à un ensemble de trajectoires du système par la relation de C-concordance et le masque observable du système. Mis à part l'efficacité du moteur de reconnaissance de chroniques, le point-clé pour la mise au point d'un système de diagnostic à base de chroniques réside dans la sélection, la spécification d'une *base de chroniques optimales*. La question qui est ainsi posée est la suivante : quels sont les critères d'une bonne, d'une mauvaise chronique ?

4.2.3 Critères de qualité

Dans [31], on propose ainsi la définition d'un ensemble de critères sur la qualité des chroniques ainsi que les moyens automatiques de vérifier ces critères sur une chronique ou un ensemble de chroniques donné.

4.2.3.1 Cohérence d'une chronique

Le premier de ces critères est la cohérence d'une chronique.

Définition 4.11 (Chronique cohérente) *Une chronique C est dite cohérente s'il existe au moins une séquence observable pouvant être reconnue par cette chronique.*

La définition de cohérence ne fait pas entrer en jeu un quelconque système. La cohérence est une notion propre à une chronique et peut se détecter en analysant simplement ses contraintes. Cette analyse s'appuie entre autres sur l'analyse du *graphe de distance* $\mathcal{G}_d = (\mathcal{X}, \mathcal{A}_d)$. Les nœuds du graphe correspondent aux variables temporelles de la chronique. Chaque arc $i \longrightarrow j$ associé à $T_{ij} = [a_{ij}, b_{ij}]$ dans la chronique est transformé en un arc $i \longrightarrow j$ associé à l'inéquation linéaire $x_j - x_i \leq c_{ij}$ où $c_{ij} = b_{ij}$ et un arc $j \longrightarrow i$ associé à l'inéquation linéaire $x_i - x_j \leq c_{ij}$ où $c_{ij} = -a_{ij}$. Le graphe de distance permet ainsi de définir la notion de chemin entre nœuds (un chemin entre un nœud i et un nœud j étant une séquence de nœuds $x_i = x_{k_0}, x_{k_1}, \ldots, x_{k_m} = x_j$ de $\mathcal{G}_d$ où $k_p \longrightarrow k_{p+1}$, $p \in \{0, m-1\}$ est un arc $\mathcal{G}_d$). Chaque chemin de longueur m entre i et j peut être associé à une distance : $x_j - x_i \leq \sum_{l=1}^{m} c_{k_{l-1}k_l}$. Et ainsi, il existe entre un nœud x_i et un nœud x_j la notion de *longueur du chemin le plus court* [Floyd, 1962], noté d_{ij} telle que $x_j - x_i \leq d_{ij}$ qui est établie par intersection de toutes les contraintes $x_j - x_i \leq \sum_{l=1}^{m} c_{k_{l-1}k_l}$ de tous les chemins de $\mathcal{G}_d$ menant de x_i à x_j. Un cycle est un chemin non-vide de $\mathcal{G}_d$ menant de x_i à x_i, il est dit négatif si $\sum_{l=1}^{m} c_{k_{l-1}k_l} < 0$.

Proposition 4.4 (Cohérence d'une chronique positive) *Une chronique positive est cohérente ssi son STP temporel sous-jacent est cohérent, à savoir, il n'existe pas de cycle négatif dans son graphe de distance.*

La non-cohérence d'une chronique peut également venir du fait que les contraintes d'interdiction sont trop restrictives si bien que l'ensemble des solutions est vide.

Définition 4.12 (Contrainte d'interdiction cohérente) *La contrainte d'interdiction d'une chronique est cohérente si elle n'empêche pas l'instanciation d'une ou plusieurs variables temporelles de la chronique.*

Théorème 4.1 *Soit la contrainte d'interdiction $(x_i, x_j, \alpha, \beta) \in \mathcal{F}(v)$, telle qu'il existe une variable temporelle x_k, où $\mathcal{M}(x_k) = v$. La contrainte d'interdiction sera cohérente si et seulement si $0 \notin [d_{ki} + \alpha, d_{kj} + \beta] \vee 0 \notin [-d_{ik} + \alpha, -d_{jk} + \beta]$.*

Le théorème 4.1 s'appuie sur le théorème de la décomposabilité des *STP* [Dechter et al., 1991] qui garantit que l'affectation d'une valeur à une variable temporelle compatible avec les contraintes locales est globalement cohérente (elle fait partie d'au moins une solution pour la chronique).

Les deux résultats précédents conduisent ainsi au moyen de vérifier si une chronique est cohérente.

Théorème 4.2 *Une chronique $\mathcal{C}$ est cohérente si et seulement si son triplet sous-jacent $(\mathcal{X}, \mathcal{A}, \mathcal{T})$, et l'ensemble des contraintes d'interdiction associées $\mathcal{F}$ sont cohérents.*

4.2.3.2 Minimalité d'une chronique

Une chronique est un ensemble de contraintes que le moteur de reconnaissance doit prendre en compte. Chaque contrainte induit une part de complexité algorithmique dans le processus de reconnaissance, aussi est-il intéressant de minimiser ce nombre de contraintes dès lors que cela est possible. À chaque chronique cohérente, une chronique minimale avec les mêmes propriétés de cohérence peut être associée (l'ensemble des séquences sur laquelle la chronique minimale est reconnue est la même que sur la première chronique). Pour obtenir la chronique minimale d'une chronique, il faut construire le graphe minimal de contraintes du *STP* sous-jacent qui est obtenu de la façon suivante. À partir du graphe des distances d'un *STP*, on construit le graphe direct qui va contenir entre tous nœuds i et j les distances des plus courts chemins d_{ij} et d_{ji}. La contrainte du graphe minimal entre i et j devient alors l'intervalle $T_{ij} = [-d_{ji}, d_{ij}]$ (où $T_{ji} = [-d_{ij}, d_{ji}]$ si $d_{ji} \geq -d_{ij}$).

Définition 4.13 (Chronique minimale) *Une chronique est minimale si son triplet sous-jacent $(\mathcal{X}, \mathcal{A}, \mathcal{T})$ est cohérent et minimal.*

La chronique minimale associée à $\mathcal{C}_1$ est illustrée à droite sur la figure 4.3.

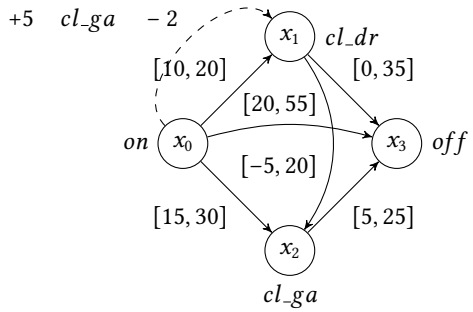


FIGURE 4.3 – Chronique minimale de $\mathcal{C}_1$.

4.2.3.3 Délai de reconnaissance d'une chronique

Le nombre d'instances d'une chronique en cours de reconnaissance (voir la figure 4.1) peut vite croître si la séquence d'entrée est longue et que la chronique est peu contrainte. Il peut donc être intéressant de prendre en compte un critère de durée de la reconnaissance pour sélectionner une chronique plutôt qu'une autre, lorsque plusieurs chroniques représentent la même situation à reconnaître. Cela peut conduire à un résultat de diagnostic au plus tôt mais également répondre à une contrainte de certains moteurs de reconnaissance qui par souci d'optimisation suppriment toutes les instances en cours de reconnaissance au delà d'un certain délai fixé.

Définition 4.14 (Durées minimale/maximale d'une chronique) *La durée minimale (resp. maximale) d'une chronique est la durée minimale (resp. maximale) entre le début de son instanciation et sa reconnaissance complète.*

Sans prendre en compte les contraintes d'interdiction, il est assez simple de calculer la durée maximale et minimale d'une chronique en se basant sur le STP sous-jacent $(\mathcal{X}, \mathcal{A}, \mathcal{T})$ minimal et complet. En effet, à un STP peuvent être associées deux solutions particulières $S_1 = (d_{01}, \dots, d_{0n})$ et $S_2 = (-d_{10}, \dots, -d_{n0})$ avec S_1 qui attribue à chaque variable la solution au plus tard, et S_2 la solution au plus tôt. Il est donc possible de déterminer D_{max_0} la durée maximale et D_{min_0} la durée minimale d'une chronique par $D_{max_0} = \max(d_{ij}, \forall i, j)$ et $D_{min_0} = \max(-d_{ji}, \forall i, j)$. Pour la chronique $\mathcal{C}_1$ (figure 4.2), $D_{max_0} = \max(d_{01}, d_{02}, d_{03}, d_{10}, \dots, d_{32}) = 55$ et $D_{min_0} = \max(-d_{01}, -d_{02}, -d_{03}, -d_{10}, \dots, -d_{32}) = 20$.

Dans le cas où les contraintes d'interdiction existent, il faut les prendre en compte. En effet, ce n'est pas parce que toutes les variables temporelles sont instanciées qu'une contrainte d'interdiction est levée. Le temps de reconnaissance peut par conséquent être rallongé, de même que la durée minimale.

Considérons les contraintes d'interdiction $(x_m, x_n, \alpha, \beta) \in \mathcal{F}(v)$ (non influentes). Le problème est de vérifier si la borne $(x_n + \beta)$ peut être supérieure à la durée maximale D_{max_0}. Notons i_{max}, i_{min} tel que $D_{max_0} = d_{i_{max}j}$ et $D_{min_0} = -d_{ji_{min}}$ (dans l'exemple précédent $i_{max} = 0$ et $i_{min} = 0$). Le calcul de la durée maximale se fait par comparaison entre D_{max_0} et $d_{i_{max}n}$ où $d_{i_{max}n}$ correspond à la longueur du chemin le plus court entre $x_{i_{max}}$ et x_n où $(x_n + \beta)$ marque la fin de la contrainte d'interdiction. La variable temporelle $x_{i_{max}}$ est prise pour origine et correspond à la variable instanciée au plus tôt. La durée minimale est obtenue de manière similaire : $D_{max} = \max(D_{max_0}, d_{i_{max}n} + \beta, \forall n)$ et $D_{min} = \max(D_{min_0}, -d_{ni_{min}} + \beta, \forall n)$. Pour la chronique $\mathcal{C}_1$ avec $\alpha = 5$ et $\beta = 35$, $D_{max} = \max(45, 20 + 35) = 55$ et $D_{min} = \max(20, 10 + 5) = 20$.

4.2.3.4 Chroniques équivalentes

Les critères précédents sont liés à la définition de la chronique en elle-même. Les deux critères suivants comparent des chroniques deux à deux.

Définition 4.15 (Équivalence) *Deux chroniques C et C' sont dites équivalentes $C \equiv C'$, si C et C' admettent les mêmes solutions quel que soit le flux d'événements observables considéré.*

Autrement dit, pour n'importe quel système, et n'importe quelle trajectoire de ce système, soit les deux chroniques sont reconnues, soit aucun d'entre elles ne l'est. Il n'y a aucun intérêt à conserver *dans la même base de chroniques* deux chroniques équivalentes, l'une d'entre elle suffit.

Théorème 4.3 *Soit C et C' deux chroniques, si leurs STPs sous-jacent $(\mathcal{X}, \mathcal{A}, \mathcal{T})$ minimaux et les contraintes d'interdiction sont identiques alors $C \equiv C'$.*

Supposons que la chronique C_1 soit modifiée en une chronique C_1' pour surveiller qu'à la suite d'une activation du clignotant droit (cl_dr) le moteur s'arrête entre 0 et 65 minutes (voir la figure 4.4). Les chroniques minimales de ces deux chroniques sont en fait identiques : $C_1 \equiv C_1'$.

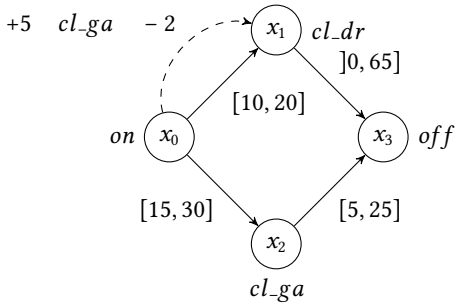


FIGURE 4.4 – Équivalence de chroniques : C_1' est équivalente à C_1.

4.2.3.5 Couverture de chroniques

La notion de couverture est importante et permet de conclure lorsqu'une chronique est reconnue qu'une autre le sera également. Dans la base de chroniques, si une chronique en couvre une autre il faut s'interroger sur la pertinence de garder les deux chroniques.

Définition 4.16 (Couverture) *Une chronique C' couvre totalement une chronique C : $C \sqsubseteq C'$ si quelles que soient les séquences S reconnues par C, elles sont également reconnues par C'.*

La notion de couverture est une notion intrinsèque aux chroniques qui ne dépend pas des séquences produites par un quelconque système. Le test que nous proposons repose sur l'hypothèse que les variables temporelles de la chronique C portent chacune un type d'événement différent ; on suppose également qu'il en est de même pour la chronique C'. On construit tout d'abord les chroniques minimales C_{min} et C'_{min}. La première étape consiste à ne pas considérer les contraintes d'interdiction.

Proposition 4.5 *C' couvre C si :*

1. *toutes les variables temporelles de C' portent des types d'événements existant dans certaines variables temporelles de C ;*
2. *pour tout arc (x'_i, x'_j), $i \neq j$, de C'_{min} il existe (x_i, x_j) dans C_{min} tel que x_i et x'_i portent le même type d'événement, de même pour x_j et x'_j et $\mathcal{T}_{ij} \subseteq \mathcal{T}'_{ij}$.*

Si les conditions précédentes sont vérifiées, il faut alors considérer, s'il y en a, les contraintes d'interdiction. Il faut s'assurer qu'une solution de C est aussi solution de C' et que les contraintes d'interdictions de C' ne bloquent pas cette solution. Pour cela, une chronique C_{couv} est mise en place à partir de C_{min} en supprimant les contraintes d'interdiction et en intégrant les contraintes d'interdiction de C' sur les événements correspondants. Si les contraintes d'interdiction de la chronique C_{couv} sont influentes sur une des variables temporelles portant un type d'événement existant sur une variable temporelle de C', alors C' ne couvre pas C puisque au moins une contrainte d'interdiction l'en empêche, du moins pour certaines solutions du STP sous-jacent de C.

Considérons la base constituée des chroniques C_2 et C_1. La chronique minimale de C_1 est représentée sur la figure 4.3 et la chronique (minimale) C_2 est présentée sur la figure 4.5. Si on ne considère pas la contrainte d'interdiction de C_2 alors $C_1 \sqsubseteq C_2$. Par contre si on considère la contrainte d'interdiction de C_2 alors dans la chronique C_{couv}, on découvre que la contrainte d'interdiction *bloque* de potentielles solutions d'affectations pour la variable x_1, C_2 ne couvre pas C_1 dans ce cas.

4.2.4 Chroniques exclusives et diagnosticabilité

Le dernier critère qui est présenté ici est le critère d'exclusivité entre chroniques, critère qui met en lien une base de chroniques et la diagnosticabilité sous-jacente du système surveillé.

4.2.4.1 Diagnosticabilité du point de vue d'une approche à base de chroniques

L'indisponibilité d'un modèle de connaissance dit des principes premiers sur un système temporel est à l'origine du développement des méthodes de diagnostic à base de chroniques : dans cette approche, on part du principe qu'il existe un expert qui est en

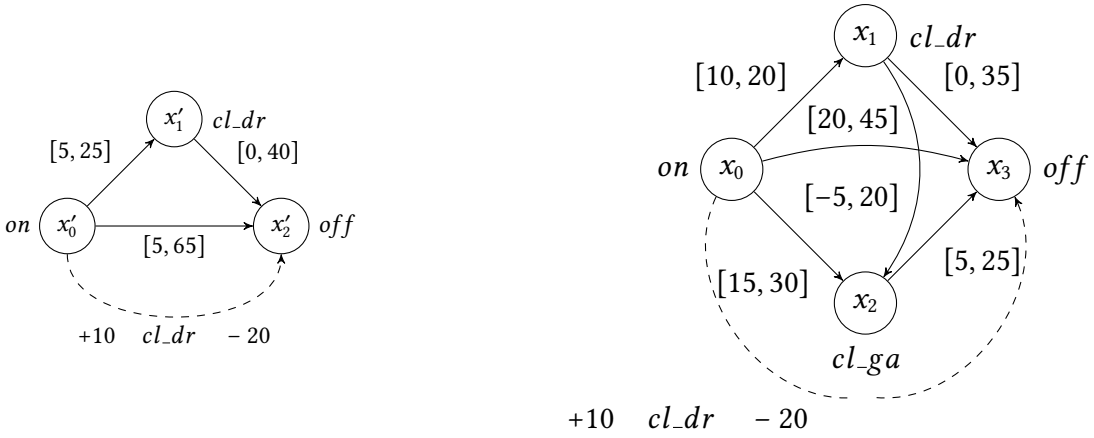


FIGURE 4.5 – Chroniques C_2 et C_{couv}.

mesure de caractériser des symptômes (chroniques) de défaillance. La question qui est posée ici est relative à cette association entre une chronique et la défaillance du système qu'elle reconnaît. Que peut-on déduire d'un ensemble de chroniques étiquetées par des défaillances en terme de diagnosticabilité sur le système sous-jacent ? À l'opposé des analyses de diagnosticabilité qui ont été présentées jusqu'à présent (voir les chapitres précédents), la connaissance de système est partielle, néanmoins il est possible de caractériser sur la base de chroniques disponibles des conditions sur la diagnosticabilité sous-jacente du (ou des) système(s) sur le(s)quel(s) la base de chroniques peut être exploitée.

Considérons donc un système temporel $\mathcal{S}$ conforme à la définition 4.4 et un motif temporel μ sur ce système (définition 4.6). Soit $\mathcal{C}(\mu)$ l'ensemble des chroniques disponibles qui sont associées à l'occurrence de μ sur le système en question. Cette association est en générale une connaissance experte qui peut avoir plusieurs sens, soit une chronique $\mathcal{C} \in \mathcal{C}(\mu)$:

1. *association de possibilité* : si la chronique est reconnue il est possible que μ ait eu lieu, autrement dit, quand μ se produit les observations caractérisées par $\mathcal{C}$ sont *possibles* ;
2. *association de certitude* : si la chronique est reconnue il est certain que μ a eu lieu, autrement dit, les observations caractérisées par $\mathcal{C}$ ne se produisent que lorsque μ a lieu.

La relation entre une base de chroniques et la diagnosticabilité sous-jacente d'un système s'appuie sur le deuxième type d'associations qui est formalisé par la notion de chronique caractéristique.

Définition 4.17 (Chronique caractéristique [49]) *Une chronique $\mathcal{C} \in \mathcal{C}(\mu)$* caractérise *$\mu$ sur un système $\mathcal{S}$ si pour toute séquence σ observable $\mathcal{C}$-concordante issue*

de $\mathcal{S}$, toutes les trajectoires τ de $\mathcal{S}$ telles que $obsT^*(\tau) = \sigma$ sont μ-concordantes.

Une chronique caractéristique garantit, si elle est reconnue, que le motif μ a effectivement eu lieu mais ne garantit pas que μ n'a pas eu lieu si elle n'est pas reconnue. En effet, μ peut être la cause de plusieurs manifestations observables différentes.

Définition 4.18 (Ensemble couvrant de chroniques [49]) *Un ensemble E de chroniques couvre μ si pour toute trajectoire τ du système $\mathcal{S}$ μ-concordante, il existe au moins une chronique C de E telle que $obsT^*(\tau)$ est C-concordante.*

La diagnosticabilité d'un système est une propriété qui garantit que les observations produites par le système sont suffisantes pour déterminer avec certitude qu'un motif μ a eu lieu. On peut ainsi présenter une condition suffisante sur la base de chroniques pour garantir qu'un motif est diagnosticable.

Proposition 4.6 ([49]) *Soit μ un motif temporel d'intérêt sur un système $\mathcal{S}$, s'il existe un ensemble de chroniques $C(\mu)$ tel que*

1. *$C(\mu)$ couvre μ ;*
2. *toute chronique de $C(\mu)$ caractérise μ ;*

alors μ est diagnosticable.

La proposition précédente exprime un lien entre une approche à base de chroniques et la diagnosticabilité sous-jacente d'un système. Il suffit en effet de déterminer un ensemble couvrant de chroniques caractéristiques pour s'assurer que le motif μ est diagnosticable et donc par conséquent, ce motif μ pourra être diagnostiqué avec certitude avec le jeu de chroniques disponibles.

4.2.4.2 Chroniques exclusives : une condition nécessaire pour la diagnosticabilité

Supposons à présent que nous ne disposons pas du modèle $\mathcal{S}$ du système mais d'un simple jeu de chroniques associées à une information de diagnostic, est-il possible d'en déduire des informations sur la qualité de ce jeu de chroniques vis-à-vis de la diagnosticabilité sous-jacente du système ? Prenons par exemple deux chroniques C_μ et $C_{\neg\mu}$, la première étant associée à l'occurrence de μ et l'autre à sa non-occurrence ($C_{\neg\mu}$ est associée à des scénarios du système où μ n'a pas (encore) eu lieu). Pour que C_μ puisse être une chronique caractéristique de μ, il faut nécessairement être assuré que C_μ ne puisse être reconnue *en même temps* que $C_{\neg\mu}$, autrement dit les chroniques C_μ et $C_{\neg\mu}$ doivent être exclusives. Considérons maintenant une autre situation où l'on dispose de deux chroniques C_{μ_1} et C_{μ_2}, s'il existe une trajectoire du système pour laquelle C_{μ_1} et C_{μ_2} sont reconnues en même temps alors on a manifestement un problème de discriminabilité

sur l'occurrence des motifs μ_1 et μ_2 qui devient même un problème de diagnosticabilité si on sait que μ_1 et μ_2 ne peuvent se produire tous les deux sur la même trajectoire du système.

Définition 4.19 (Exclusivité) *Deux chroniques C et C' sont dites exclusives si elles ne peuvent être reconnues sur un support temporel commun par un même flux d'événements.*

La propriété d'exclusivité que nous proposons est intrinsèque à un couple de chroniques, elle est indépendante d'un quelconque système et peut donc être analysée de façon indépendante au même titre que l'équivalence ou la couverture (voir section 4.2.3). Cette propriété est importante pour le diagnostic car elle est en lien avec la diagnosticabilité et/ou la discriminabilité des motifs d'intérêt sur un système donné. Par support temporel commun on entend ici que le support temporel de reconnaissance (intervalle de temps entre le début et la fin de sa reconnaissance) de l'une des chroniques démarre en même temps que le support temporel de l'autre.

FIGURE 4.6 – Deux chroniques exclusives.

La figure 4.6 présente un tel couple de chroniques exclusives. La chronique C attend un événement on suivi d'un événement off dans un intervalle entre 10 et 20 unités de temps alors que C' attend un même événement on et l'absence de off entre 5 et 40.

4.2.5 Résumé

Cette section a présenté la formalisation générale des chroniques et de leur reconnaissance. Cette formalisation a permis de définir notamment un lien formel entre la méthode générale de reconnaissance de chroniques et le problème de diagnostic temporel que l'on cherche à résoudre. Une chronique C est un motif observable capable de représenter implicitement l'ensemble des trajectoires du système qui sont C-concordantes. À partir de cette formalisation, on a mis en évidence des critères de qualité et de comparaisons entre chroniques afin de mettre en évidence la nécessité de *filtrer* dans la base de chroniques existante les chroniques les plus pertinentes à considérer pour la reconnaissance. Cette section a également mis en évidence des liens entre la qualité d'une base de chroniques et la diagnosticabilité sous-jacente d'un système. Ce lien nécessite d'analyser en particulier la propriété d'exclusivité entre chroniques. Cette analyse nécessite l'utilisation d'un formalisme à partir duquel

il est possible de faire de la vérification automatique de propriétés (*model checking*) [Clarke et al., 1999, Schnoebelen, 1999, Baier and Katoen, 2008]. Notre proposition est le sujet de la section suivante.

4.3 Analyses de diagnosticabilité et chroniques

4.3.1 Contexte des travaux

Cette section porte sur les analyses de diagnosticabilité qui ont été développées à l'aide d'un formalisme particulier de réseau de Petri. Le développement de ces différentes analyses a débuté en collaboration avec Audine Subias dans le cadre du projet européen WS-DIAMOND. Cette collaboration s'est ensuite poursuivi par le co-encadrement du stage de Master 2 (chroniques exclusives) et de la thèse (diagnosticabilité de motifs temporels) de Houssam-Eddine Gougam.

4.3.2 Formalisme d'analyse

Les analyses qui ont été développées dans le cadre des systèmes temporels nécessitent des techniques de vérification de modèle. À cette fin, nous avons orienté notre choix vers un outil de vérification qui a été développé au LAAS-CNRS : l'outil TINA (*TIme petri Net Analyzer*) [Berthomieu et al., 2004]. Ainsi, le formalisme utilisé est une extension des réseaux de Petri (voir la section 3.3.5.1) au temps : il s'agit des réseaux de Petri temporels à priorité [Bérard et al., 2005, Berthomieu et al., 2007].

Définition 4.20 (RPTEPL) *Un réseau de Petri temporel étiqueté à priorité de type L (acronyme RPTEPL) est un octuplet* $\langle P, T, A, \succ, \ell, \Sigma, I_s, Q\rangle$ *tel que :*

- *le septuplet* $\langle P, T, A, \succ, \ell, \Sigma \cup \{\lambda\}, Q\rangle$ *est un RPEPL (voir la définition 3.27), où* $\ell : T \longrightarrow \Sigma \cup \{\lambda\}$ *est la fonction d'étiquetage de transition avec* λ *l'étiquette d'un événement temporel (voir définition 4.2);*
- $I_s : T \longrightarrow \mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \{\infty\})$ *est la fonction d'intervalle temporel statique qui associe à chaque transition un intervalle de temps sur* $\mathbb{R}^+$ *ou un intervalle tel que* $[a, \infty[\subseteq \mathbb{R}^+$ *ou* $]a, \infty[\subseteq \mathbb{R}^+$.

Un RPTEPL marqué est un RPTEPL muni d'un marquage initial M_0.

Un *état* du RPTEPL est un couple $E = \langle M, I\rangle$ où M est un marquage et I est une application de fenêtre temporelle de tir :

$$I : T \longrightarrow \mathbb{R}^+ \times (\mathbb{R}^+ \cup \{\infty\})$$

qui associe à chaque transition t du RPTEPL un intervalle temporel dans lequel la transition *doit* être déclenchée si elle est sensibilisée (la sensibilisation est identique à celle décrite dans la section 3.3.5.1). Une transition t est *tirable* à partir d'un état $E = \langle M, I\rangle$ à une date relative δ si et seulement si :

1. t est sensibilisée par M, et
2. $\delta \geq \lfloor I(t) \rfloor$ si $I(t)$ est fermé à gauche, ou $\delta > \lfloor I(t) \rfloor$ si $I(t)$ est ouvert à gauche, et
3. pour toute transition t' sensibilisée par M, la date δ est plus petite que la date de tir la plus tardive de t' : $\delta \leq \lceil I(t') \rceil$ si $I(t)$ est fermé à droite, ou $\delta < \lceil I(t') \rceil$ si $I(t)$ est ouvert à droite.

Ainsi, pour être tirable, une transition doit être non seulement sensibilisée mais elle doit aussi respecter des contraintes temporelles. Deux transitions t et t' sont dites *en conflit* sur un marquage M si elles sont toutes les deux sensibilisées mais $\exists p \in \text{pre}(t) \cap \text{pre}(t'), M(p) = 1$. Tirer une transition t à une date relative δ de l'état $E = \langle M, I \rangle$ conduit à l'état $E' = \langle M', I' \rangle$ tel que :

— $M' = M \setminus \text{pre}(t) \cup \text{post}(t)$;
— Pour toute transition $t' \in T$:

1. si t' n'est pas sensibilisée par M', $I'(t')$ est un intervalle vide (par exemple $]1, 1]$) ;
2. si t' est distincte de t et était sensibilisée par M sans être en conflit avec t pour M, soit $[]$ une notation représentant l'un des symboles [ou] en fonction des cas :
   — si $I(t') = [a, b[]$ alors $I'(t') = [max(0, a - \delta), b - \delta[]$;
   — si $I(t') =]a, b[]$ alors $I'(t') =]a - \delta, b - \delta[]$ si $\delta \leq a$, $I'(t') = [0, b - \delta[]$ sinon ;
   — si $I(t') = [a, +\infty[$ alors $I'(t') = [max(0, a - \delta), +\infty[$;
   — si $I(t') =]a, +\infty[$ alors $I'(t') =]a - \delta, +\infty[$ si $\delta \leq a$, $I'(t') = [0, +\infty[$ sinon.
3. sinon $I'(t') = I_s(t')$.

Un tir de transition à la date δ est noté : $\langle M, I \rangle \xrightarrow{\delta t} \langle M', I' \rangle$. Avec les règles de tir définies précédemment, on peut ainsi définir des séquences de transitions tirables et la notion d'états atteignables. L'état initial du RPTEPL est $E_0 = \langle M_0, I_0 \rangle$ tel que $I_0(t) = I_s(t), \forall t \in T$.

Considérons maintenant une telle séquence τ, $E_0 \xrightarrow{\tau} E$, on peut lui associer la séquence temporelle $\delta\ell_{t\hat{o}t}(\tau)$:

— si τ est la séquence vide alors $\delta\ell_{t\hat{o}t}(\tau) = 0\lambda$;
— si $\tau = \delta_1 t_1 \delta_2 t_2 \dots \delta_k t_k, k > 0$ alors $\delta\ell_{t\hat{o}t}(\tau) = \delta_1 \ell(t_1) \delta_2 \ell(t_2) \dots \delta_k \ell(t_k)$.

$\delta\ell_{t\hat{o}t}(\tau)$ est la séquence temporelle qui se termine exactement au temps du dernier tir de transition. Le système peut alors rester dans ce marquage pendant un temps non nul $d \in [0, a[]$ où a peut être fini ou non, si bien que la séquence de transitions τ est également associée aux séquences temporelles $\{\delta\ell_{t\hat{o}t}(\tau).d\lambda, d \in [0, a[]\}$. Soit $\delta\ell(\tau)$ l'ensemble des séquences temporelles canoniques équivalentes à l'ensemble $\{\delta\ell_{t\hat{o}t}(\tau).d\lambda, d \in [0, a[]\}$, on peut alors caractériser le langage temporel généré par un RPTEPL.

Définition 4.21 (Langage d'un RPTEPL marqué) *Le langage généré par le RPTEPL marqué* $N = \langle P, T, A, \succ, \ell, \Sigma, I_s, Q, M_0 \rangle$ *est :*

$$\mathcal{L}(N) = \bigcup_{\tau : E_0 \xrightarrow{\tau} E = \langle M, I \rangle \wedge M \in Q} \delta\ell(\tau) \subseteq \mathcal{T}(\Sigma).$$

4.3.3 Analyse de l'exclusivité de chroniques

La première analyse qui a été développée est celle de l'exclusivité des chroniques (voir la section 4.2.4). Cette analyse nécessite dans un premier temps de traduire une chronique en un RPTEPL.

4.3.3.1 Traduction des chroniques en RPTEPL

L'objectif a été de déterminer un moyen systématique de traduire toute chronique en un RPTEPL à partir duquel il est possible de déterminer si deux chroniques sont effectivement exclusives ou non. Dans [23] [1], on propose une telle traduction. Le principe est de décrire la chronique comme la composition d'éléments atomiques. Ces éléments atomiques sont ainsi composés en séquence (un élément suivant l'autre), en divergence (un élément suivi d'un ensemble d'éléments en parallèle), en convergence (un ensemble d'éléments en parallèle suivi d'un élément). Pour illustrer cette traduction, on considère la chronique suivante (le détail théorique est exposé dans [23]) :

```
chronicle exempleTrad
{
   event(a,t1); event(b,t2); event(d,t3); event(c,t4);
   noevent(e,t5); event(e,t6);
   t1+1 <= t2 <= t1+2;
   t1 <= t3 <= t1+3;
   t2+2 <= t4 <= t2+4;
   t3+1 <= t4 <= t3+10;
   t1+2 <= t5 < t1+4;
   t5+3 <= t6 <= t5+6;
}
```

Cette chronique est simple mais illustre les différents éléments atomiques et leurs combinaisons. Il existe ici 6 éléments atomiques (les six prédicats). La séquence est illustrée par les prédicats `event(c,t4)` et `event(e,t6)`, l'événement `e` suit en séquence l'événement `c`. La divergence est illustrée ici par l'événement `a` qui est suivi en parallèle des événements `b` et `d`. La convergence est quant à elle illustrée par les

1. Le travail de traduction décrit ici est antérieur à la modélisation formelle qui a été présentée en section 4.2.2, une mise à jour prenant en compte cette nouvelle formalisation reste à mettre en place.

événements parallèles `b` et `d` qui sont suivis de l'événement `c`. La chronique dispose également d'un prédicat `noevent`. La traduction de cette chronique en RPTEPL est présentée sur la figure 4.7. Chaque prédicat est associé à des contraintes temporelles. Le prédicat `event(a,t1)` n'a pas de contrainte, cet élement est représenté sur la figure par le sous-réseau de Petri constitué des places p_1 et p_3. À l'inverse, l'élément `event(e,t6)` est conditionné dans $[3, 6]$ après `event(c,t5)`. Cet élément est représenté par les places p_{14} (l'événement `c` vient de survenir, on initialise l'horloge pour l'apparition de `e`), p_{15} (3 unités de temps sont passées, on entre en phase de reconnaissance de `e`), p_{16} (l'événement `e` a eu lieu avant 3 unités de temps donc entre 3 et 6 après `c`). L'utilisation de priorité entre la transition $\lambda]3, +\infty[$ et $e[0, +\infty[$ est un moyen de garantir que toute transition étiquetée avec un événement non-temporel (comme ici l'événement e) n'est pas temporellement contrainte. Cette propriété est nécessaire pour la synchronisation des réseaux de Petri, opération qui est nécessaire pour l'analyse de l'exclusivité (voir la section suivante). La divergence est représentée quant à elle par la transition $\lambda[0, 0]$ entre les places p_3, p_6 et p_7. Elle permet de mettre en parallèle les événements `b` et `d` qui suivent `a`. Le prédicat associé à `b` est représenté par les places p_6, p_8, p_{10}, et celui associé à `d` est représenté par les places p_7, p_9, p_{11}. La convergence est mise en œuvre par une transition $c[0, +\infty[$ qui resynchronise les horloges issues des occurrences de `b` et `d`. Cette synchronisation est possible car c n'est pas contraint temporellement, les contraintes qui lui sont associées sont portées par des priorités. Enfin le prédicat `noevent(e,t5)` est représenté par les places p_2, p_4, p_5. Un jeton n'est présent dans p_5 que si `e` n'a pas eu lieu entre 2 et 4 après `a`. Le RPTEPL n'a qu'un marquage final : $Q = \{\{p_5, p_{16}\}\}$.

4.3.3.2 Méthode d'analyse

Analyser l'exclusivité de deux chroniques $\mathcal{C}_1$ et $\mathcal{C}_2$ consiste d'abord à traduire celles-ci en deux RPTEPLs marqués $\mathcal{N}_1$ et $\mathcal{N}_2$ avec la méthode présentée dans la section précédente et ensuite à construire un nouveau RPTEPL résultat d'une opération produit spécifique entre $\mathcal{N}_1$ et $\mathcal{N}_2$. Le principe de ce produit consiste à capturer l'ensemble des comportements possibles d'un système quelconque pouvant être reconnus par chacune des deux chroniques et de vérifier s'il existe au moins un comportement pour lequel les deux chroniques sont reconnues sur le même support temporel. Le produit en question, noté $\mathcal{N}_1 \bowtie \mathcal{N}_2$ est défini de la façon suivante.

Définition 4.22 *Soit $\mathcal{N}_1$ le RPTEPL $\langle P_1, T_1, A_1, \succ_1, \ell_1, \Sigma_1, I_{s1}, Q_1, M_{01}\rangle$ issu de la traduction de la chronique $\mathcal{C}_1$, soit $\mathcal{N}_2$ le RPTEPL $\langle P_2, T_2, A_2, \succ_2, \ell_2, \Sigma_2, I_{s2}, Q_2, M_{02}\rangle$ issu de la traduction de la chronique $\mathcal{C}_2$, le produit $\mathcal{N} = \mathcal{N}_1 \bowtie \mathcal{N}_2$ est le RPTEPL $\langle P, T, A, \succ, \ell, \Sigma, I_s, Q\rangle$ tel que :*

— $P = P_1 \cup P_2$;

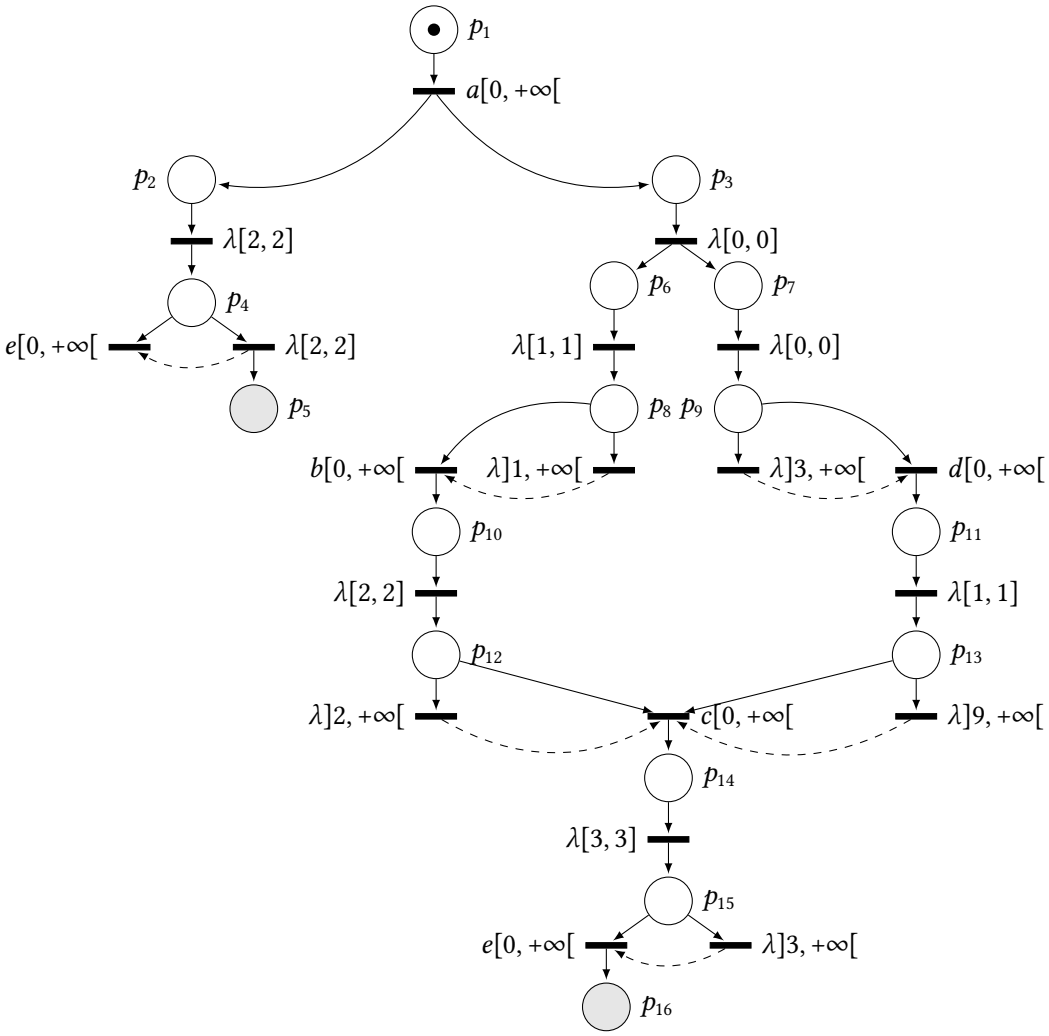


FIGURE 4.7 – Traduction de la chronique `exempleTrad` en RPTEPL.

— $T = T_1 \cup T_2 \cup T_s$ *où* T_s *sont les transitions synchronisées :*

$$T_s = \bigcup_{e \in \Sigma_1 \cap \Sigma_2} \{t_1 \| t_2, t_1 \in T_1, t_2 \in T_2, \ell_1(t_1) = e, \ell_2(t_2) = e\}.$$

— $A = A_1 \cup A_2 \cup A_s$ *tel que*

$$\begin{aligned} A_s = &\{(p,t) : p \in P_1, t = t_1 \| t_2, (p,t_1) \in A_1\} \\ &\cup \{(p,t) : p \in P_2, t = t_1 \| t_2, (p,t_2) \in A_2\} \\ &\cup \{(t,p) : p \in P_1, t = t_1 \| t_2, (t_1,p) \in A_1\} \\ &\cup \{(t,p) : p \in P_2, t = t_1 \| t_2, (t_2,p) \in A_2\}. \end{aligned}$$

— $\succ = \succ_1 \cup \succ_2 \cup \succ_s$,

$$\begin{aligned} \succ_s = &\{(t_s,t_1),(t_s,t_2) : t_s = t_1 \| t_2 \in T_s\} \cup \\ &\bigcup_{t_s = t_1 \| t_2 \in T_s} \{ \bigcup_{t_i \in \{t_1,t_2\}} \{(t_s,t) : (t_i,t) \in \succ_i\} \cup \\ &\qquad \bigcup_{t_i \in \{t_1,t_2\}} \{(t,t_s) : (t,t_i) \in \succ_i\}\}. \end{aligned}$$

— $\forall t = t_1 \| t_2 \in T_s, \ell(t) = \ell_1(t_1), \forall t \in T \cap T_1, \ell(t) = \ell_1(t), \forall t \in T \cap T_2, \ell(t) = \ell_2(t)$.
— $\Sigma = \Sigma_1 \cup \Sigma_2$.
— $\forall t \in T, I_s(t) = I_{s1}(t)$ *si* $t \in T_1, I_s(t) = I_{s2}(t)$ *si* $t \in T_2, I_s(t) = [0, +\infty[$ *sinon.*
— $Q = \{q_1 \cup q_2 : (q_1,q_2) \in Q_1 \times Q_2\}$.
— $M_0 = M_{01} \cup M_{02}$.

La figure 4.8 illustre ce produit sur les deux chroniques $\mathcal{C}$ et $\mathcal{C}'$ de la figure 4.6. Le principe est de confronter les deux chroniques en créant une transition synchronisée $t_1 \| t_2$ pour tout couple de transitions (t_1, t_2) où chaque transition appartient à une chronique différente et est étiquetée par le même type d'événement $e \neq \lambda$. Cette nouvelle transition $t_1 \| t_2$ traduit la reconnaissance commune de l'événement $e = \ell_1(t_1) = \ell_2(t_2) = \ell(t_1 \| t_2)$. Ainsi chaque événement e d'un flux quelconque peut ainsi être considéré comme faisant partie de la reconnaissance de l'une, de l'autre, voire des deux chroniques simultanément en fonction des cas. La transition synchronisée est définie comme prioritaire sur t_1 et t_2. Si les deux transitions t_1 et t_2 sont effectivement tirables à un moment donné dans les réseaux initiaux alors seule la transition $t_1 \| t_2$ l'est dans $\mathcal{N}_1 \bowtie \mathcal{N}_2$.

La vérification de l'exclusivité va se faire ensuite à l'aide de l'outil `selt` de TINA (TIme petri Net Analyzer) selon le même principe que celui utilisé pour l'étude de la diagnosticabilité de motif atemporel (voir la section 3.3.5.4). TINA se charge de calculer à partir du produit $\mathcal{N}_1 \bowtie \mathcal{N}_2$ la structure de Kripke correspondante et `selt` de vérifier la propriété souhaitée. La propriété souhaitée est issue du résultat suivant.

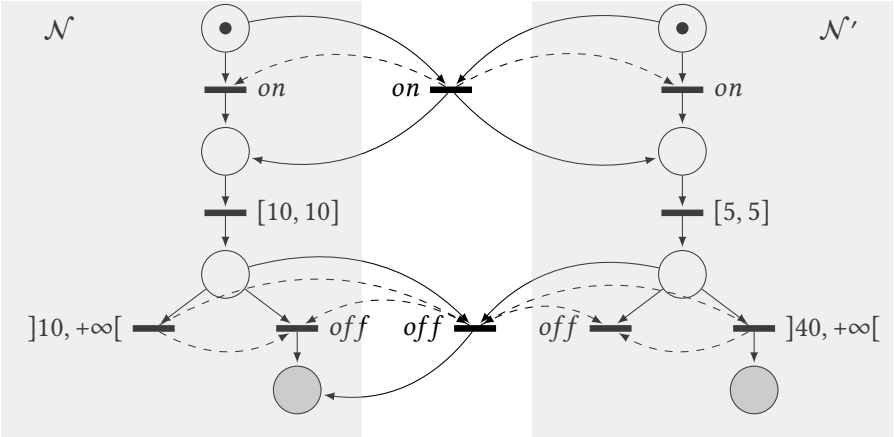


FIGURE 4.8 – Produit des chroniques exclusives de la figure 4.6 : $\mathcal{N} \bowtie \mathcal{N}'$ fondé sur la traduction de $\mathcal{C}$ en $\mathcal{N}$ et de $\mathcal{C}'$ en $\mathcal{N}'$.

Théorème 4.4 ([49]) *Les chroniques sont exclusives si et seulement si le langage* $\mathcal{L}(\mathcal{N} \bowtie \mathcal{N}') = \emptyset$.

Autrement dit, les deux chroniques sont exclusives si les marquages de Q définis dans $\mathcal{N} \bowtie \mathcal{N}'$ ne sont pas accessibles, ce qui se traduit dans `selt` par la formule SE-LTL suivante [Chaki et al., 2004] :

$$\varphi_{\text{EXCLUSIVE}} = \forall\Box\Diamond\neg(M \in Q).$$

La formule $\varphi_{\text{EXCLUSIVE}}$ dit que pour toute trajectoire de $\mathcal{N} \bowtie \mathcal{N}'$ ($\forall$), il est toujours vrai ($\Box$) que cette trajectoire aboutira ($\Diamond$) à ce que le marquage courant ne soit pas accepteur ($\neg(M \in Q)$). Dans le cas de la figure 4.8, il n'existe aucune exécution de ce réseau de Petri qui va conduire à marquer les deux places grisées du seul marquage accepteur de ce réseau. Pour que les chroniques $\mathcal{C}$ et $\mathcal{C}'$ ne soient pas exclusives, il faut trouver deux supports temporels qui démarrent au même instant, à savoir au moment où un événement *on* apparaît. La transition synchronisée de *on* garantit que la reconnaissance des deux chroniques a bien démarré en même temps. Le problème maintenant est que *off* est attendu entre 10 et 20 par $\mathcal{C}$ alors qu'il est rejeté par $\mathcal{C}'$ entre 5 et 40. Un tel couple de support n'existe donc pas, les chroniques sont exclusives.

4.4 Méthode de détection et de localisation de défauts sur des graphes d'événements temporisés

4.4.1 Contexte des travaux

La présente section a pour objectif de présenter le problème de diagnostic temporel sur une classe de systèmes temporisés particulière, celle pouvant être représentée par des graphes d'événements temporisés. Ce travail a été effectué dans le cadre d'une collaboration interne avec Euriell Le Corronc et la supervision du stage de Master 2 d'Alexandre Sahuguède et du stage de Master 1 de Claire Paya.

4.4.2 Systèmes (max,+) linéaires

Dans cette contribution, nous nous intéressons essentiellement à des systèmes tels que des lignes d'assemblages, des chaînes automatisées de production. Dans ce type de système, on dispose d'un ensemble de machines qui effectuent en parallèle des activités. Chaque machine exploite en entrée des ressources et le résultat de son activité est un produit qui peut servir comme ressource pour une autre machine. Le résultat de ce réseau de machines est la production, à partir d'un flot de ressources en entrée (matière première, éléments de base), d'un flot de produits finis (produits assemblés et prêts pour la livraison client). Pour la modélisation de ce type de système, nous adoptons le point de vue suivant. Chaque machine démarre une activité en fonction des ressources disponibles à une date donnée (représentée par un événement de départ deb) et termine son activité après une durée d (représentée par un événement de fin fin) en ayant consommé les ressources d'entrée et en ayant produit de nouvelles ressources disponibles pour d'autres machines. Adoptant ce point de vue, il est intuitif de modéliser ce type de système par une sous-classe de réseau de Petri que l'on nomme les graphes d'événements temporisés (acronyme GET).

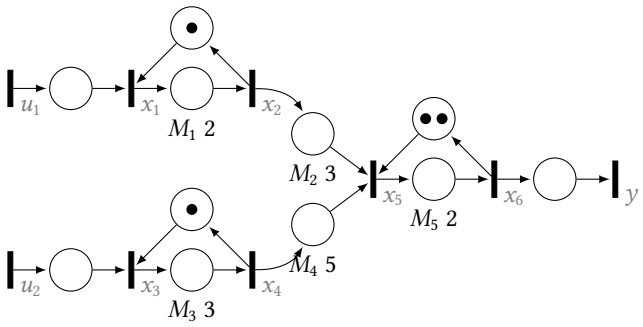


FIGURE 4.9 – Un GET d'une ligne d'assemblage.

Définition 4.23 (Graphe d'événements temporisés) *Un* graphe d'événements temporisés *est un réseau de Petri* (P, T, A, δ) *où le triplet* (P, T, A) *constitue un réseau de Petri classique et* $\delta : P \longrightarrow \mathbb{R}^+$ *modélise un graphe d'événements : toute place de P a exactement une transition d'entrée et une transition de sortie. La fonction* δ *associe à chaque place une durée.*

Un tel GET est représenté sur la figure 4.9. Les places p ne contenant aucune durée sont telles que $\delta(p) = 0$. La notion de durée δ modifie les conditions de sensibilisation d'une transition de Petri. En effet, pour que la place p appartenant à un ensemble $pre(t)$ participe à la sensibilisation de t il faut que le jeton soit dans la place p pendant la durée $\delta(p)$. Les conditions de franchissement quant à elles, sont restreintes *aux conditions de franchissement au plus tôt.*

Définition 4.24 (Franchissement au plus tôt) *Un GET a un fonctionnement dit au plus tôt dès lors que toute transition t (telle que* $pre(t) \neq \emptyset$*) est franchie à sa date effective de sensibilisation.*

La figure 4.9 représente ainsi une ligne d'assemblage. Les transitions u_1 et u_2 représentent l'arrivée des ressources sur la chaîne. Ce système n'est pas autonome dans le sens où son comportement est conditionné par l'arrivée des ressources à des dates données. Supposons par exemple que deux ressources soient disponibles sur u_1 et sur u_2 à la même date δ et qu'aucune autre ressource n'est disponible par la suite, alors, en reprenant la notation définie dans la section 4.1.2, le système peut générer la séquence :

$$\delta u_1 0 u_2 0 x_1 0 x_3 2 x_2 1 x_4 5 x_5 2 x_6 0 y$$

Une fois la ressource disponible sur u_1, l'activité de la machine M_1 démarre instantanément par le franchissement de la transition x_1 représentant l'événement deb de la machine M_1. L'activité de la machine M_1 dure $\delta(M_1) = 2$ unités de temps et se termine par l'occurrence de x_2 à la date de fin de l'activité de M_2. En parallèle, la machine M_3 démarre une activité entre les événements x_3 et x_4 dont la durée est de 3 unités de temps. L'événement x_5 est un événement de synchronisation des machines M_2 et M_4. x_5 n'est franchie qu'à la date où les deux machines M_2 et M_4 ont effectivement terminé leur activité. M_2 ayant commencé à la date 2, elle a terminé à la date 5, par contre la machine M_4 n'a démarré qu'à la date 3 et a donc terminé son activité à la date 8. L'événement x_5 a donc lieu à la date $1 + 2 + 5 = 8$ qui démarre ainsi l'activité de la machine M_5 qui effectue l'assemblage des produits issus des machines M_2 et M_4.

Dans ces travaux, le formalisme utilisé n'est pas de type GET. La classe des réseaux GET est une restriction particulière des réseaux de Petri temporels sur laquelle il est possible d'exploiter un formalisme dans lequel les problèmes de diagnostic posés ici vont reposer sur une représentation *linéaire* du système grâce à un système algébrique :

une algèbre dite $(max, +)$. Plus précisément, nous proposons dans ces travaux de modéliser le système à l'aide du dioïde $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ dont la définition nécessite des étapes intermédiaires (nous ne présentons ici que les éléments essentiels à la compréhension de notre contribution en diagnostic, une présentation rigoureuse de cette théorie peut se trouver dans [Baccelli et al., 1992].

Définition 4.25 (Dioïde) *Un dioïde (ou anneau semi-idempotent) est un ensemble $\mathcal{D}$ muni de deux opérations $\oplus$ et $\otimes$ avec les propriétés suivantes.*

1. *Associativité de l'addition :* $\forall a, b, c \in \mathcal{D}, (a \oplus b) \oplus c = a \oplus (b \oplus c)$.
2. *Commutativité de l'addition :* $\forall a, b \in \mathcal{D}, a \oplus b = b \oplus a$.
3. *Associativité de la multiplication :* $\forall a, b, c \in \mathcal{D}, (a \otimes b) \otimes c = a \otimes (b \otimes c)$.
4. *Distributivité de la multiplication sur l'addition :* $\forall a, b, c \in \mathcal{D}, (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ *et* $c \otimes (a \oplus b) = c \otimes a \oplus c \otimes b$.
5. *Existence d'un élément neutre :* $\exists \varepsilon \in \mathcal{D} \ : \ \forall a \in \mathcal{D}, a \oplus \varepsilon = a$.
6. *Élément neutre absorbant :* $\forall a \in \mathcal{D}, a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon$.
7. *Existence d'un élément identité :* $\exists e \in \mathcal{D} \ : \ \forall a \in \mathcal{D}, a \otimes e = e \otimes a = a$.
8. *Idempotence de l'addition :* $\forall a \in \mathcal{D}, a \oplus a = a$.

Le dioïde que l'on va exploiter est également *complet*, autrement dit il est fermé pour les sommes infinies et le produit $\otimes$ distribue sur ces sommes infinies. On considère également qu'il est *commutatif* pour la multiplication ($a \otimes b = b \otimes a$). Dans la suite de ce document, l'opérateur de multiplication $\otimes$ sera omis dans les expressions ($(a \otimes b) = ab$). La notation a^i correspondra au produit $aa \ldots a$, i fois. Ainsi $a^0 = e$. Enfin, on notera l'opérateur de Kleene comme suit :

$$\forall a \in \mathcal{D}, a^\star = \bigoplus_{i \geq 0} a^i.$$

L'indempotence de la loi additive induit une relation d'ordre sur cette structure algébrique qui forme donc également un treillis. La relation d'ordre $\preceq$ est définie comme suit :

$$\forall a, b \in \mathcal{D}, a \preceq b \equiv a \oplus b = b.$$

Dans le formalisme proposé, l'ensemble des exécutions possibles du système est représenté à l'aide de séries formelles. Ces séries sont formées à l'aide de deux variables commutatives :

— γ représente une occurrence d'événement,
— δ représente une durée.

Les séries formelles que nous exploitons ont la forme canonique suivante :

$$s = \bigoplus_{n,t\in\mathbb{Z}} s(n,t)\gamma^n\delta^t$$

avec $s(n,t) = e$ ou ε. L'ensemble de ces séries forment alors un dioïde complet qui est noté $\mathbb{B}[\![\gamma,\delta]\!]$. Considérons par exemple la série suivante :

$$u_1 = \gamma^0\delta^2 \oplus \gamma^1\delta^3 \oplus \gamma^2\delta^5 \oplus \gamma^4\delta^{+\infty}.$$

Si nous associons u_1 à la transition u_1 du GET de la figure 4.9 alors la série u_1 représente un tirage possible de la transition u_1 :

- $\gamma^0\delta^2$: le 1er tirage de u_1 se produit à l'instant 2 ;
- $\gamma^1\delta^3$: le 2ème tirage de u_1 se produit à l'instant 3 ;
- $\gamma^2\delta^5$: le 3ème tirage de u_1 se produit à l'instant 5 ;
- et il n'y a pas de 4ème tirage.

Il est maintenant à noter que les séries que nous cherchons à exploiter ont une forme particulière à savoir que le nombre d'occurrences d'événement ainsi que le temps vont croissant (on note la série $\gamma^{i_1}\delta^{j_1} \oplus \cdots \oplus \gamma^{i_2}\delta^{j_2}$ telle $i_1 < i_2$ et $j_1 < j_2$). L'ensemble de ces séries forment un sous-dioïde de $\mathbb{B}[\![\gamma,\delta]\!]$, c'est le dioïde $\mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]$ (le dioïde $\mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]$ est formellement défini comme l'ensemble quotient $(\gamma \oplus \delta^{-1})^\star\mathbb{B}[\![\gamma,\delta]\!]$).

À l'aide de $\mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]$, il est ainsi possible de représenter un GET sous la forme d'un système matriciel d'équations d'états :

$$\begin{cases} X = AX \oplus BU \\ Y = CX \end{cases}$$

où U, X et Y sont respectivement les vecteurs d'entrée, d'états et de sortie du système ($U \in \mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]^q, X \in \mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]^n, Y \in \mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]^p$) et $A \in \mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]^{n\times n}, B \in \mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]^{n\times q}, C \in \mathcal{M}_{in}^{ax}[\![\gamma,\delta]\!]^{p\times n}$ sont des matrices de séries formelles.

Dans le cas de la figure 4.9, le modèle est le suivant. Le vecteur d'états X est un vecteur représentant par une variable toute transition du GET qui n'est pas une transition d'entrée ou de sortie. Les vecteurs U et Y représentent quant à eux les transitions d'entrées et de sorties. Les points dans les matrices remplacent l'élément neutre ε.

$$\begin{cases} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} . & \gamma^1\delta 0 & . & . & . & . \\ \gamma^0\delta^2 & . & . & . & . & . \\ . & . & . & \gamma^1\delta_0 & . & . \\ . & . & \gamma^0\delta^3 & . & . & . \\ . & \gamma^0\delta^3 & . & \gamma^0\delta^5 & . & \gamma^2\delta^0 \\ . & . & . & . & \gamma^0\delta^2 & . \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \oplus \begin{bmatrix} \gamma^0\delta^0 & . \\ . & . \\ . & \gamma^0\delta^0 \\ . & . \\ . & . \\ . & . \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \\ \begin{bmatrix} y \end{bmatrix} = \begin{bmatrix} . & . & . & . & . & \gamma^0\delta^0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \end{cases}$$

À titre d'exemple, la transition x_5 est associée à l'équation d'état :

$$x_5 = \gamma^0\delta^3 x_2 \oplus \gamma^0\delta^5 x_4 \oplus \gamma^2\delta^0 x_6$$

Cette équation exprime *la solution au plus tôt* pour laquelle le tirage de x_5 a lieu au moins 3 unités de temps suivant le tirage de x_2, au moins 5 unités de temps suivant le tirage de x_4 et au moins 2 transitions x_6 ont été tirées précédemment.

4.4.3 Définition des indicateurs de décalages temporels par résiduation

Dans le cadre de ces travaux, le système temporel observé (voir la définition 4.5) est constitué du triplet $(\mathcal{S}, \Sigma_o, obsT)$ où :

- $\mathcal{S}$ rassemble l'ensemble des trajectoires possibles du GET, représentées comme les solutions en séries formelles du système d'équations d'états définis dans $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$.
- Σ_o contient l'ensemble des étiquettes de transitions u, y correspondant aux transitions d'entrée et de sortie du GET.
- $obsT$ projette les trajectoires de $\mathcal{S}$ sur l'ensemble des événements u et y et sur leur date d'occurrence.

La question de diagnostic qui est posée dans un premier temps est la suivante. Sachant que l'on observe les entrées u et les sorties y du système, peut-on déterminer si le système se comporte correctement, à savoir si le comportement observé est cohérent avec le comportement prédit.

Afin de calculer le comportement prédit pour un scénario d'entrée u donné, on va exploiter le fait qu'à partir de la représentation du système en équations d'états dans $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$, il est possible de calculer une *fonction matricielle de transfert* H qui établit un lien direct entre les entrées U du système et les sorties Y prédites par son modèle. La fonction de transfert H est obtenue à partir du système d'équation d'états par l'équation suivante :

$$H = CA^{\star}B$$

si bien que l'on peut exploiter par la suite la relation entrée-sortie :

$$Y = HU.$$

La matrice H constitue ainsi un modèle fonctionnel liant la sortie du système avec son entrée. Reprenant l'exemple de la figure 4.9, la fonction de transfert est la suivante :

$$H = \begin{bmatrix} \gamma^0\delta^7(\gamma^1\delta^2)^{\star} & \gamma^0\delta^{10}(\gamma^1\delta^3)^{\star} \end{bmatrix}.$$

La prédiction du comportement du système est alors établie en calculant $\tilde{Y}$ à partir de H et de U, par exemple si

$$U = \begin{bmatrix} \gamma^0\delta^2 \oplus \gamma^1\delta^3 \oplus \gamma^2\delta^5 \oplus \gamma^4\delta^{+\infty} \\ \gamma^0\delta^2 \oplus \gamma^1\delta^3 \oplus \gamma^2\delta^5 \oplus \gamma^4\delta^{+\infty} \end{bmatrix},$$

alors

$$\tilde{Y} = HU = \begin{bmatrix} \gamma^0\delta^{12} \oplus \gamma^1\delta^{15} \oplus \gamma^2\delta^{18} \oplus \gamma^3\delta^{21} \oplus \gamma^4\delta^{+\infty} \end{bmatrix}.$$

Revenons un instant sur le lien entre le modèle temporel $(S, \Sigma_o, obsT)$ issue la définition 4.5 et la représentation que l'on utilise ici. La matrice U contient le scénario d'entrée, elle exprime implicitement toutes les trajectoires (au plus tôt) du GET, qui, une fois projetées sur les entrées, donnent sur l'exemple les trajectoires suivantes (aux permutations près entre les u_1 et u_2 consécutifs) :

$$2u_1 0 u_2 1 u_1 0 u_2 2 u_1 0 u_2 d\lambda, d \geq 0.$$

Un événement de type u_1 a lieu à deux unités de temps, de même qu'un événement de type u_2 suivis des occurrences de u_1 et de u_2 après 1 unité de temps, etc. De même, en projetant sur les sorties ces mêmes trajectoires produisent :

$$12y3y3y3yd\lambda, d \geq 0.$$

Autrement dit, les trajectoires du GET qui sont considérées par l'équation $\tilde{Y} = HU$ sont les trajectoires τ qui, selon le masque observable $obsT$ défini ici, sont telles que :

$$obsT(\tau) = 2u_1 0 u_2 1 u_1 0 u_2 2 u_1 0 u_2 7 y3y3y3yd\lambda, d \geq 0.$$

L'étape suivante va alors consister à établir une comparaison entre la sortie prédite $\tilde{Y}$ et la sortie observée Y et va donc constituer le test de cohérence du diagnostic. Ce test de cohérence va mettre à profit les possibilités de calcul dans $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ et notamment exploiter la théorie de la résiduation.

Dans [61], la solution proposée se focalise sur les *décalages temporels*. L'observation des sorties Y du système produit une séquence temporelle d'événements. Sur l'exemple de la figure 4.9, qui est constitué d'une seule sortie y, on peut par exemple observer :

$$12y3y4y4yd\lambda, d \geq 0$$

ce qui peut se traduire par une série temporelle sur y :

$$Y = \left[y\right] = \left[\gamma^0\delta^{12} \oplus \gamma^1\delta^{15} \oplus \gamma^2\delta^{19} \oplus \gamma^3\delta^{23} \oplus \gamma^4\delta^{+\infty}\right]$$

Ainsi la comparaison entre l'observation réelle du système et la prédiction revient toujours à la comparaison de deux séries formelles y et $\tilde{y}$ (dans le cas de notre exemple mono-sortie $Y = \left[y\right]$ et $\tilde{Y} = \left[\tilde{y}\right]$). Les deux séries sont formellement de la forme :

$$y = \bigoplus_{n\in\mathbb{Z}} \gamma^n\delta^{\mathcal{D}_y(n)}, \tilde{y} = \bigoplus_{n\in\mathbb{Z}} \gamma^n\delta^{\mathcal{D}_{\tilde{y}}(n)}$$

où $\mathcal{D}_y(n)$ et $\mathcal{D}_{\tilde{y}}(n)$ sont des fonctions dateurs [30].

Définition 4.26 (Fonction de décalage temporel) *Soient $y, \tilde{y} \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ et leurs fonctions dateurs respectives, la fonction de décalage temporel est définie par*

$$\mathcal{T}_{y,\tilde{y}}(n) = \mathcal{D}_{\tilde{y}}(n) - \mathcal{D}_y(n).$$

La fonction de décalage mesure l'écart temporel des deux séries (notamment on a $\forall n \in \mathbb{Z}, \mathcal{T}_{y,y}(n) = 0$). Intuitivement, il suffit donc de déterminer un indice n tel que $\mathcal{T}_{y,\tilde{y}}(n) \neq 0$ pour affirmer que le comportement observé n'est pas normal. Pour faire cette comparaison, on utilise la théorie de la résiduation. Cette théorie étudie les inéquations $f(x) \preceq y$ et plus particulièrement, en ce qui nous concerne, les inéquations de forme $yx \preceq \tilde{y}$ et $\tilde{y}x \preceq y$ sur $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$. Dans cette théorie, la fonction f est dite *résiduable* s'il existe au moins une solution à $f(x) \preceq y$. De plus, comme $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ est complet, il existe toujours dans ce cas une plus grande solution au sens de $\preceq$ et l'application $f^{\#}$ qui associe à x cette plus grande solution est l'application résiduée de f. Dans notre cas d'étude, les applications résiduées que nous exploitons retournent les plus grandes solutions (les plus grands résidus) de $yx \preceq \tilde{y}$ et $\tilde{y}x \preceq y$, elles sont définies comme suit :

$$y\not\backslash\tilde{y} = \bigoplus\{x : x\tilde{y} \preceq y\}, \tilde{y}\not/y = \bigoplus\{x : xy \preceq \tilde{y}\}$$

Intuitivement, on va donc établir les séries résiduelles $y\not\backslash\tilde{y}$ et $\tilde{y}\not/y$ à partir desquelles il sera possible d'extraire l'information recherchée sur $\mathcal{T}_{y,\tilde{y}}(n)$. Cette information nous est donnée par le théorème suivant [MaxPlus, 1991] :

Théorème 4.5 *Soient* $y, \tilde{y} \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$,

$$\forall n \in \mathbb{Z}, \mathcal{D}_{\tilde{y}y}(0) \leq \mathcal{T}_{y,\tilde{y}}(n) \leq -\mathcal{D}_{y\tilde{y}}(0)$$

Notons $\Sigma_\tau(y, \tilde{y})$ l'intervalle $[\mathcal{D}_{\tilde{y}y}(0), -\mathcal{D}_{y\tilde{y}}(0)]$. La borne inférieure de $\Sigma_\tau(y, \tilde{y})$ correspond au décalage minimum dans le domaine temporel entre y et $\tilde{y}$ alors que la borne supérieure correspond au décalage maximum. Le théorème 4.5 indique donc que la comparaison entre les séries y et $\tilde{y}$ peut être réduite à déterminer les bornes $\mathcal{D}_{\tilde{y}y}(0)$ et $-\mathcal{D}_{y\tilde{y}}(0)$ de la fonction de décalage temporel $\mathcal{T}_{y,\tilde{y}}$. Ces bornes sont extraites des séries $y \not\circ \tilde{y}$ et $\tilde{y} \not\circ y$. Plus précisément :

— $\mathcal{D}_{\tilde{y}y}(0)$ provient du monôme $\gamma^0\delta^{\mathcal{D}_{\tilde{y}y}(0)} \in \tilde{y} \not\circ y$.

— $-\mathcal{D}_{y\tilde{y}}(0)$ provient du monôme $\gamma^0\delta^{\mathcal{D}_{y\tilde{y}}(0)} \in y \not\circ \tilde{y}$.

De ceci découle finalement l'indicateur de décalage temporel :

Définition 4.27 (Indicateur de décalage temporel [61]) *Soit H la matrice de transfert d'un système $(max, +)$-linéaire. Soient $U = \left[u_i\right]_i \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]^p$ et $Y = \left[y_i\right]_i \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]^q$ les observations du système, l'indicateur $I_H(U, y_i)$ est la fonction :*

$$I_H(U, y_i) = \begin{cases} faux & \text{si } \tilde{Y} = HU \text{ et } \tilde{y}_i = \tilde{Y}[i] \text{ et } \Sigma_\tau(y_i, \tilde{y}_i) = [0, 0] \\ vrai & \text{sinon} \end{cases}$$

Proposition 4.7 ([61]) *L'indicateur $I_H(U, y_i)$ est correct, il ne retourne $vrai$ que si le comportement du système est anormal.*

Ainsi, on peut associer à chaque sortie y_i un indicateur de cette forme et dès lors que l'un des indicateurs retourne *vrai*, la détection d'un décalage temporel entre le comportement prédit et le comportement observé est ainsi confirmé. Si nous reprenons l'exemple où

$$y = \gamma^0\delta^{12} \oplus \gamma^1\delta^{15} \oplus \gamma^2\delta^{19} \oplus \gamma^3\delta^{23} \oplus \gamma^4\delta^{+\infty}$$

et

$$\tilde{y} = \gamma^0\delta^{12} \oplus \gamma^1\delta^{15} \oplus \gamma^2\delta^{18} \oplus \gamma^3\delta^{21} \oplus \gamma^4\delta^{+\infty}$$

on constate que :

$$y \not\circ \tilde{y} = \gamma^{\mathbf{0}}\delta^{\mathbf{0}} \oplus \gamma^1\delta^3 \oplus \gamma^2\delta^7 \oplus \gamma^3\delta^{11} \oplus \gamma^4\delta^{+\infty}$$

et que :

$$\tilde{y} \not\circ y = \gamma^{\mathbf{0}}\delta^{-2} \oplus \gamma^1\delta^2 \oplus \gamma^2\delta^6 \oplus \gamma^3\delta^9 \oplus \gamma^4\delta^{+\infty}.$$

L'indicateur $I_H(U, y)$ du GET de la figure 4.9 va donc établir un intervalle $\Sigma_\tau(y, \tilde{y}) = [-2, 0]$ et ainsi retourner la détection d'un décalage temporel.

4.4.4 Localisations des sources de décalage

Les travaux de [61] proposent ainsi un moyen de déterminer si le système se comporte normalement ou non à partir de l'observation des entrées-sorties. La deuxième étape proposée dans [30] consiste à aller plus loin dans l'analyse en localisant les origines possibles de ces sources de dysfonctionnement. Les motifs d'intérêt (voir la définition 4.6 page 120) qui sont considérés ici sont des décalages d'événements. Reprenons par exemple le cas de la figure 4.9. La machine M_5 démarre une opération à l'instant où x_5 est tirée et s'arrête à l'instant où x_6 est tirée et dans le comportement normal du système x_6 a lieu à 2 unités de temps de x_5. Une faute de type décalage temporel sur M_5 correspond ainsi au motif temporel $\{ dx_5^i d' x_6^i \ : \ d \geq 0, d' \neq 2, i \in \mathbb{N} \}$ où x_j^i dénote la i-ème occurrence de x_j. Dans le cadre de la représentation dans le dioïde $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$, ces décalages temporels vont être modélisés par des monômes de faute.

Définition 4.28 (Monôme de faute) *Un monôme de faute est un monôme de la forme :*

$$m_i = \gamma^{n_i} \delta^{t_i} \otimes f_i$$

avec

$$f_i = \begin{cases} \gamma^0 \delta^{t_i'} & \textit{si une faute de type décalage est présente} \\ e & \textit{sinon.} \end{cases}$$

Par exemple, un décalage de 3 unités de temps sur la mcahine M_5 sera représenté par un monôme

$$m_1 = \gamma^0 \delta^2 \otimes \gamma^0 \delta^3 = \gamma^0 \delta^{2+3} = \gamma^0 \delta^5$$

qui décale le tir de x_6 de 3 unités de temps.

L'effet d'une faute de type décalage temporel peut être constaté sur plusieurs sorties et donc concerner plusieurs indicateurs. À partir de la fonction de transfert écrite explicitement à l'aide des monômes de faute m_i, il est possible d'obtenir les liens de causalité entre les fautes possibles et leurs effets sur les indicateurs. L'analyse de cette fonction de transfert conduit ainsi à la mise en place d'un ensemble de signatures pour toute faute f_i. Si un indicateur $I_H(U, y_j)$ *peut* retourner vrai avec l'occurrence d'un décalage f_i alors toute combinaison des indicateurs telles que $I_H(U, y_j)$ est vrai constitue un fait observable appartenant à la signature de f (au sens de la définition 2.14 page 34). Par construction du modèle, on peut également déterminer s'il existe des indicateurs $I_H(U, y_j)$ qui *doivent* retourner vrai dès que f_i se produit. Dans ce cas, un tel indicateur à vrai fait partie de tout fait observable de la signature de f_i.

4.4.5 Résumé

Les travaux présentés ici introduisent une méthode pour la résolution du problème de diagnostic temporel (détection, localisation) pour une sous-classe de systèmes : les GETs. Dans cette sous-classe, on bénéficie d'un formalisme de calcul offrant une représentation linéaire du système à partir de laquelle, on peut aisément définir un ensemble d'indicateurs formels et corrects pouvant ainsi participer à la détection et à la localisation de dysfonctionnements provoquant des décalages temporels dans un système type ligne d'assemblage. Ce premier travail théorique a donné lieu à la mise en œuvre de la plateforme logicielle de diagnostic (max,+) développée par Claire Paya et qui s'appuie sur la bibliothèque `minmaxgd` de [Cottenceau et al., 2000].

4.5 Résumé

L'observation du temps dans les systèmes à événements discrets apporte une information très significative pour raffiner le résultat du diagnostic. Analyser le temps dans un raisonnement de diagnostic nécessite de gérer une difficulté supplémentaire à savoir la continuité du temps. Outre le caractère combinatoire qui est déjà présent dans le diagnostic de SED atemporel, l'ajout de la mesure du temps peut mener dans certains cas à des problèmes d'indécidabilité. Les travaux présentés ici autour de cette thématique porte sur une analyse en amont des propriétés du système pour déterminer si un algorithme de diagnostic de SED temporel aura de bonnes performances. L'algorithme de prédilection pour le diagnostic de SED temporel est un algorithme de reconnaissance de chroniques. Ce choix est lié essentiellement aux collaborations scientifiques à travers les projets. Le premier travail a consisté à formaliser la notion de chroniques en vue d'y adjoindre la notion de contraintes d'interdiction. La présence de contraintes sur l'absence d'événements a un intérêt essentiel en diagnostic car il offre un moyen puissant de discriminer les chroniques, de les rendre notamment exclusives. La deuxième contribution porte sur l'analyse automatique de propriétés sur les chroniques afin d'améliorer la base de chroniques en terme de performance, diagnosticabilité. Notamment, afin de mettre en place la vérification de l'exclusivité des chroniques, une traduction automatique des chroniques en RPTEPL a été mis en place, ce qui a permis de mettre à profit le vérificateur de modèle TINA pour cette analyse. Enfin, des travaux plus récents ont défini le problème de diagnostic temporel sur une sous classe de réseau de Petri, les Graphes d'Événements Temporisés afin de bénéficier de méthodes de calcul qui reposent sur une représentation linéaire du système.

Chapitre 5

Diagnostic et ses applications à la décision

Les chapitres précédents ont porté sur le diagnostic à travers la mise en place de méthodes ou d'analyses. Le but de ce chapitre est de décrire les contributions à des problèmes qui s'appuient sur un résultat de diagnostic. En général, le diagnostic d'un système n'est pas un but en soi, il constitue une étape intermédiaire à partir de laquelle on peut résoudre un problème de décision. L'objectif de ce chapitre est de présenter les liens qui ont été exploités entre le raisonnement diagnostique et d'aide à la décision. Ici, le terme décision est pris au sens large, il sera décliné sous plusieurs formes à travers la description des contributions.

5.1 Systèmes autoguérissants

5.1.1 Contexte et objectif

Le travail sur les *systèmes autoguérissants* a été mené dans le cadre du projet européen WS-DIAMOND [21, 1, 17, 2, 3, 18]. en collaboration étroite avec l'un des partenaires du consortium, l'université de Rennes 1. L'objectif de ce projet était en autre de modéliser le comportement d'un ensemble de services Web à l'aide d'un formalisme de systèmes à événements discrets [Salaün et al., 2004], [69, 70] et ainsi de mettre en évidence le lien entre le diagnostic et la décision de réparation dans ce contexte. Comme présenté dans [19], un système est autoguérissant si et seulement si, après l'occurrence d'une faute, un diagnostic peut être fait à la suite duquel un plan de réparation adapté à la situation est automatiquement déclenché. La question qui est ainsi posée est la suivante : soit un système à événements discrets (voir la section 3.1), quelles sont les conditions sur son comportement pour que le système soit autoguérissant ? La caractérisation formelle de l'autoguérison d'un système dépend de conditions sur la

capacité de diagnostic (diagnosticabilité) et de conditions sur la capacité de réparation (réparabilité). On considérera tout au long de cette étude que les motifs d'intérêts sont de simples occurrences de fautes, de plus, on considérera l'hypothèse de la faute simple, à savoir que le système ne subit qu'un seul type de faute à la fois.

5.1.2 Notion de réparation

Comme nous l'avons vu dans le chapitre 3, dans un système à événements discrets la notion d'états est définie par la séquence des événements τ issue du langage $\mathcal{S}$ qui a conduit à cet état (voir la définition 3.5). Dans un tel paradigme, si un événement de faute se produit à partir d'un état du SED, il a pour effet que tous les états qui lui succèdent, représentent des états dysfonctionnels : les comportements possibles après la faute sont considérés comme anormaux, ce n'est qu'une suite d'états dysfonctionnels (sous l'hypothèse que les effets de la faute sont permanents). Autrement dit, l'occurrence de la faute conduit le système à changer de *mode* opérationnel. Reprenant les notations logiques du chapitre 2 sur la notion de mode (voir la section 2.3.2 page 33), tous les états x successeurs de l'occurrence de f sont donc dans le mode f :

$$x \vDash \mathsf{f} \text{ ssi } x \text{ est la cible d'une trajectoire } \tau \in \mathcal{S} \wedge f \in \tau.$$

À titre illustratif, on reprend le formalisme des automates (voir la section 3.2.2.1) dans une approche centralisée (un système est représenté par un seul automate, son modèle global). La figure 5.1 représente l'automate d'un tel système. Les événements $\{f_i\}_{i\in\{1,\dots,4\}}$ sont des événements fautifs et les événements $\{o_j\}_{j\in\{1,\dots,6\}}$ sont observables. Ce système contient ainsi 5 modes opérationnels : ok, f_1 , f_2, f_3, f_4. Dans cet automate, chacun de ses états x_i est une abstraction d'un ensemble d'états du SED qui sont dans le même mode opérationnel. Par exemple, l'état x_{13} regroupe toutes les trajectoires de mode f_1 qui sont constituées de l'occurrence d'un événement f_1 suivi d'un nombre non nul de o_6 et l'état x_4 représente exactement la trajectoire o_4 qui est de mode ok. [1]

La réparation de f est intuitivement un procédé qui ramène le système d'un état x_f du mode f ($x_f \vDash \mathsf{f}$) vers un état x_{ok} du mode $x_{ok} \vDash \mathsf{ok}$ où les effets de la faute ne s'expriment plus [Friedrich et al., 1992], [Sun and Weld, 1993], [Friedrich et al., 1994], [Nejdl and Bachmayer, 1993] : [2]

$$x_{ok} \vDash \mathsf{ok} \text{ ssi } x_{ok} \text{ est la cible d'une trajectoire } \tau \in \mathcal{S} \wedge \forall f \in \Sigma_f, f \notin \tau.$$

Il est à préciser ici que la notion de réparation est caractérisée en intention. Le calcul effectif de cette réparation et le modèle sous-jacent qui lui est nécessaire définissent

1. Il faut noter qu'il existe plusieurs automates pouvant représenter le même SED, cet automate a été choisi dans cette forme à titre illustratif, afin que tout état de l'automate puisse être associé à exactement un mode.

2. Je rappelle ici que l'on est sous l'hypothèse de la faute simple, à tout instant, le système est soit dans le mode ok, f_1, f_2 ... mais jamais dans un mode tel que $\mathsf{f}_1 \wedge \mathsf{f}_2$.

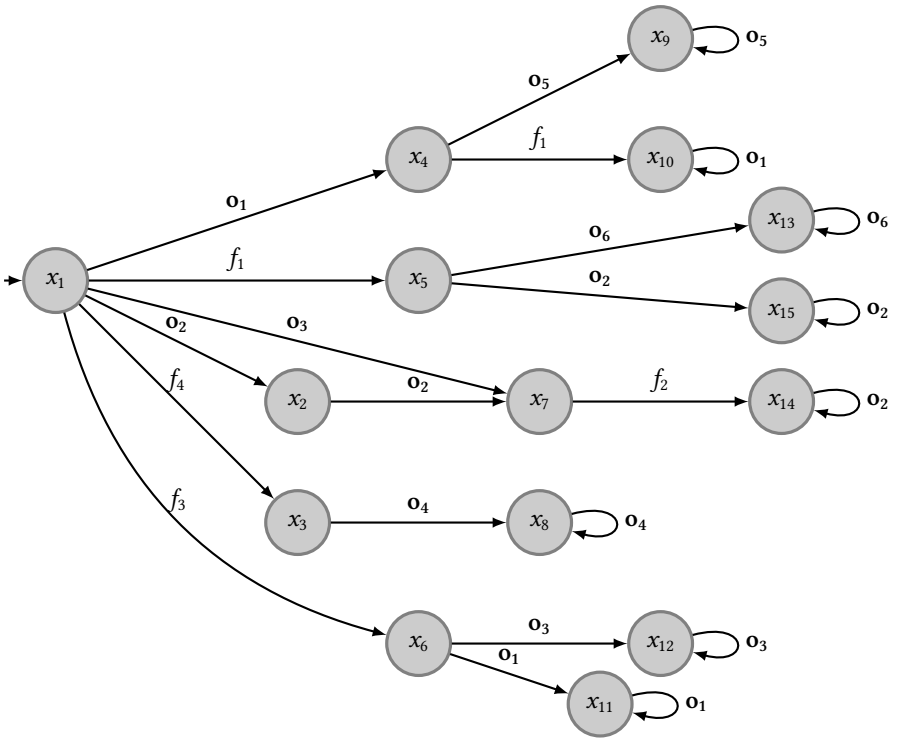


FIGURE 5.1 – Un système autoguérissant.

un problème de planification à part entière et indépendant de la contribution que nous présentons ici. En particulier, le modèle de comportement tel que celui présenté sur la figure 5.1 ne contient pas les éléments pour établir un tel plan de réparation [Ramadge and Wonham, 1989].

Définition 5.1 (Réparation) *Soit un SED de langage $\mathcal{S}$ et f un événement de faute sur ce système, soit $x_f \models \mathsf{f}$ une réparation $r_f(x_f)$ est un état du mode nominal :*

$$r_f(x_f) \models \mathsf{ok}.$$

Dans un système de type SED, on peut aisément imaginer qu'un processus de réparation est caractérisé par une séquence d'événements choisie qui permet ainsi de ramener le SED fautif d'un état x de mode f à un état $r_f(x)$. Ce plan de réparation peut faire partie d'un plan multi-objectifs dont l'un serait la réparation [Washington et al., 1999].

Définition 5.2 (Plan de réparation) *Soit un SED de langage $\mathcal{S}$ et f un événement de faute sur ce système, le plan R est un plan de réparation de f s'il existe au moins un état*

$x \models \mathsf{f}$ à partir duquel l'application du plan R conduit à une réparation $r_f(x)$. Le fait que $r_f(x)$ résulte de l'application du plan R à partir de l'état x est noté

$$\mathsf{repare}(\mathsf{R}, \mathcal{S}, \mathsf{x}, \mathsf{r_f(x)}).$$

Un plan de réparation [Mayer et al., 2012] dépend en général d'un état initial (un état $\mathsf{x} \models \mathsf{f}$). Un plan de réparation R est universel s'il est un plan qui peut être appliqué dans n'importe quel état $\mathsf{x} \models \mathsf{f}$ et qui garantit d'atteindre à chaque fois un état de réparation $r_f(x)$ (plan conformant au sens de [Smith and Weld, 1998]). Un plan de réparation universel pour le mode f est représenté par le prédicat suivant :

$$\mathsf{repare_f}(\mathsf{R}, \mathcal{S}) \text{ ssi}$$

$$\text{pour tout } \mathsf{x} \models \mathsf{f} \text{ il existe } \mathsf{r_f(x)} \text{ tel que } \mathsf{repare}(\mathsf{R}, \mathcal{S}, \mathsf{x}, \mathsf{r_f(x)}).$$

Afin de compléter la caractérisation d'un plan de réparation à tous les modes possibles, on définit également la notion de plan de réparation pour le mode nominal par extension :

$$\mathsf{repare_{ok}}(\mathsf{R}, \mathcal{S}) \text{ ssi}$$

$$\text{pour tout } \mathsf{x} \models \mathsf{ok} \text{ il existe } \mathsf{r(x)} \text{ tel que } \mathsf{repare}(\mathsf{R}, \mathcal{S}, \mathsf{x}, \mathsf{r(x)}).$$

On peut notamment remarquer que le plan vide $\mathsf{R}_\emptyset$ (c-à-d celui tel que pour tout $\mathsf{x} \models \mathsf{ok}$, on a $\mathsf{repare}(\mathsf{R}_\emptyset, \mathcal{S}, \mathsf{x}, \mathsf{x})$) est un plan de réparation universel pour le mode nominal.

Il est à noter qu'un plan de réparation est caractérisé par un ensemble de couples (état défectueux,réparation), un plan R étant totalement caractérisé en intention par l'ensemble des prédicats vrais $\mathsf{repare}(\mathsf{R}, \mathcal{S}, \mathsf{x}, \mathsf{r_f(x)})$. Ainsi, un même plan de réparation peut réparer des fautes différentes, autrement dit, il peut réparer un macromode (voir définition 2.16). Soit le macromode $\mathsf{mf} = \mathsf{f}_{i_1} \vee \cdots \vee \mathsf{f}_{i_n}$, un plan de réparation universel R pour le macromode mf est caractérisé par :

$$\mathsf{repare_{mf}}(\mathsf{R}, \mathcal{S}) \equiv \bigwedge_{j \in 1}^{n} \mathsf{repare_{f_{ij}}}(\mathsf{R}, \mathcal{S}).$$

Définition 5.3 (Réparabilité d'un macromode mf**)** *Un macromode* mf *est réparable dans un système $\mathcal{S}$ s'il existe au moins un plan universel R de ce mode :*

$$\mathsf{reparable_{mf}}(\mathcal{S}) \equiv \exists \mathsf{R}, \mathsf{repare_{mf}}(\mathsf{R}, \mathcal{S}).$$

De même un ensemble de macromodes M est réparable dans un système $\mathcal{S}$ si :

$$\mathsf{reparable}_M(\mathcal{S}) \equiv \bigwedge_{\mathsf{mf} \in M} \mathsf{reparable_{mf}}(\mathcal{S}).$$

Cette première définition de réparabilité conduit à celle du système.

Définition 5.4 (Réparabilité d'un système) *Un système $\mathcal{S}$ est réparable si tout mode de faute* f *est réparable :*

$$\mathsf{reparable}(\mathcal{S}) \equiv \bigwedge_{i=1}^{n} \mathsf{reparable}_{\mathsf{f_i}}(\mathcal{S}).$$

La condition de réparabilité du système est ainsi réalisée si l'on est en mesure de déterminer pour tout mode de faute élémentaire, un plan de réparation universel pour ce mode. Maintenant la difficulté est liée au fait que pour appliquer un plan de réparation, il faut être en mesure de garantir que le mode de faute correspondant est bien le mode courant. L'observabilité du système peut ne pas garantir que l'on ait suffisamment d'information pour affirmer que le mode du système courant soit déterminé de telle manière que l'on puisse appliquer le bon plan de réparation. Ainsi la condition d'autoguérison souhaitée ici [Shaw, 2001], [Ghosh et al., 2007] est liée non seulement à la condition de réparabilité mais également à celle de diagnosticabilité.

5.1.3 Caractérisation d'un système autoguérissant

Soit le modèle de SED $\mathsf{DS} = (\mathcal{S}, \Sigma_o, obs)$ (voir définition 3.7 page 60), un système autoguérissant est formellement défini de la façon suivante. On suppose que le système peut subir une faute parmi un ensemble de n fautes simples $\{f_1, \ldots, f_n\}$. Sous l'hypothèse de faute simple, le système est donc à tout instant dans l'un des modes suivants : $\mathsf{ok}, \mathsf{f}_1, \ldots, \mathsf{f}_n$. Soit T un ensemble non vide de trajectoires $\mathcal{S}$, on note $\mathsf{mf}(T)$ le macromode correspondant à la disjonction des modes $\mathsf{f_i}$ où $\mathsf{f_i}$ est le mode d'au moins l'état cible d'une trajectoire de T.

Définition 5.5 (Système autoguérissant) *Un système* DS *est autoguérissant si*

$$\begin{gathered}\exists n \in \mathbb{N}, \forall \tau_1.f \text{ une séquence de } \mathcal{S}, f \in \{f_1, \ldots, f_n\}, \\ \forall \tau_2 \in \mathcal{S}/\tau_1.f \ : \ |obs(\tau_2)| \geq n \implies \\ \exists \mathsf{R} \ : \ \text{soit } T = \{\tau_3 \in \mathcal{S}, obs(\tau_3) = obs(\tau_1.\tau_2)\}, \mathsf{repare}_{\mathsf{mf(T)}}(\mathsf{R}, \mathcal{S}).\end{gathered}$$

Intuitivement, on peut remarquer que la définition de l'autoguérison est liée à la diagnosticabilité, il suffit de comparer cette définition avec la définition 3.21 page 87 pour s'en convaincre. Ce que l'on dit ici est qu'un système est autoguérissant si on peut toujours le réparer après l'occurrence d'une faute f qui est suivie d'un nombre fini d'observations. Mais il y a une subtilité, il n'est en effet pas nécessaire de garantir que le mode de faute f est effectivement le mode courant pour débuter la réparation. Dès lors que l'on est en mesure de déterminer à l'aide des observations un macromode mf réparable contenant le mode f, on peut lancer le plan de réparation et ainsi réparer la faute f.

5.1.4 Lien entre diagnosticabilité et réparabilité pour l'autoguérison

L'autoguérison établit un compromis entre la diagnosticabilité et la réparabilité. Plus les macromodes réparables sont grands, moins il est nécessaire d'avoir des conditions fortes de diagnosticabilité ; plus les macromodes réparables sont petits, plus fortes devront être les conditions de diagnosticabilité. Cette sous-section a pour but de présenter formellement ce lien.

Dans le cadre de la caractérisation d'un système autoguérissant, les fautes sont limitées à l'occurrence d'événements simples. La condition de diagnosticabilité établie dans la définition 3.21 page 87 s'applique donc ici sur des motifs simples. On définit le prédicat suivant :

$$\mathsf{diagnosticable}_{\mathsf{f}}(\mathcal{S}) \text{ ssi}$$

le motif f (associé au mode f) est diagnosticable au sens de la définition 3.21 dans le système $\mathcal{S}$. Sous l'hypothèse de la faute simple dans le système $\mathcal{S}$, il est possible de caractériser la condition de diagnosticabilité d'une autre manière qui va nous permettre de montrer par la suite comment l'autoguérison résulte de la diagnosticabilité et de la réparabilité.

Le système $\mathcal{S}$ est dans le mode f dès lors qu'il a produit une trajectoire $\tau \in \mathcal{S}$ qui contient f. Tout prolongement de τ dans $\mathcal{S}$ est donc par hypothèse dans le mode f. Dès lors que le système est vivant (voir les hypothèses de la section 3.3.2), on peut définir une trajectoire *à la limite* τ^∞ (limite d'une suite de prolongements de τ dans $\mathcal{S}$: τ, $\tau.e_1$, $\tau.e_1e_2$, ...). Cette trajectoire τ^∞ est une séquence infinie d'événements ($\tau^\infty \in \Sigma^\infty$) dans laquelle f a eu lieu : ainsi on dit que τ^∞ est de mode f. L'observabilité du système étant supposée également vivante, l'application du masque observable *obs* sur τ^∞ produit également une séquence infinie d'événements observables σ^∞ : la séquence σ^∞ est un élément de la *signature infinie* du mode f [Cordier et al., 2006], [Pucel et al., 2007].

Définition 5.6 (Signature infinie de mode) *Soit* DS *le modèle d'un système et* f *un mode de faute de* $\mathcal{S}$*, la signature infinie de* f*,* $Sig^\infty(\mathsf{f})$*, est l'ensemble des séquences infinies d'événements observables* σ^∞ *issues de l'application du masque observable sur toute trajectoire infinie* τ^∞ *de mode* f *issue de* DS*.*

Reprenant l'exemple de la figure 5.1, la signature infinie $Sig^\infty(\mathsf{f}_1)$ du mode f_1 est :

$$Sig^\infty(\mathsf{f}_1) = \{o_1^\infty, o_6^\infty, o_2^\infty\}$$

et celle du mode ok est :

$$Sig^\infty(\mathsf{ok}) = \{o_1 o_5^\infty\}.$$

À l'aide de cette notion de signature, il est possible de réécrire la diagnosticabilité d'un mode f de la façon suivante.

Proposition 5.1 *Soit* $\{\mathsf{f}_1, \ldots, \mathsf{f}_n\}$ *les modes de faute d'un système, soit* f_j *l'un de ces modes :*

$$\mathsf{diagnosticable}_{\mathsf{f}_j}(\mathcal{S}) \equiv Sig^{\infty}(\mathsf{f}_j) \cap Sig^{\infty}(\mathsf{ok}) = \emptyset \wedge$$
$$\forall \mathsf{i} \in \{1, \ldots, \mathsf{n}\} \setminus \{\mathsf{j}\}, Sig^{\infty}(\mathsf{f}_j) \cap Sig^{\infty}(\mathsf{f}_i) = \emptyset.$$

Un mode de faute est diagnosticable si sa signature infinie n'est pas en intersection avec la signature infinie d'un autre mode de faute, le mode ok inclus. Dans le cadre de l'exemple de la figure 5.1, $\mathsf{diagnosticable}_{\mathsf{f}_4}(\mathcal{S})$ est vrai car la signature de f_4 ne contient que la séquence o_4^{∞} qui est propre à la signature de f_4. Par contre, on a $\neg\mathsf{diagnosticable}_{\mathsf{f}_2}(\mathcal{S})$ car $Sig^{\infty}(\mathsf{f}_1) \cap Sig^{\infty}(\mathsf{f}_2) = \{o_2^{\infty}\} \neq \emptyset$.

Proposition 5.2 *Soit* $F = \{\mathsf{f}_1, \ldots, \mathsf{f}_n\} \cup \{\mathsf{ok}\}$ *les modes d'un système, le système est diagnosticable (*$\forall \mathsf{f} \in F, \mathsf{diagnosticable}_{\mathsf{f}}(\mathcal{S})$*) ssi* $\{Sig^{\infty}(\mathsf{f})\}_{\mathsf{f} \in F}$ *est une partition de l'ensemble des séquences infinies et observables du système.*

Ainsi le système représenté par la figure 5.1 n'est pas diagnosticable. Notamment les modes f_1 et f_2 ne sont pas diagnosticables.

La caractérisation de la diagnosticabilité d'un système s'appuie sur les modes de fautes élémentaires et nécessite que les signatures de ces modes réalisent une partition de l'ensemble des séquences infinies et observables du système. Il est également possible d'étendre cette caractérisation sur des ensembles de macromodes M. Soit Ω un ensemble non vide de séquences observables infinies du système $\mathcal{S}$, soit $\mathsf{MF}(\Omega)$ le macromode

$$\mathsf{MF}(\Omega) = \bigvee_{\mathsf{f} : \sigma \in Sig^{\infty}(\mathsf{f}) \cap \Omega} \mathsf{f}.$$

Définition 5.7 (Diagnosticabilité d'un ensemble de macromodes) *Soit* $M = \{\mathsf{mf}_1, \ldots, \mathsf{mf}_n\}$ *un ensemble de macromodes sur le système* $\mathcal{S}$*, M est un ensemble de macromodes diagnosticable (noté* $\mathsf{diagnosticable}_M(\mathcal{S})$*) s'il existe une partition* $\pi = \{\Omega_1, \ldots, \Omega_n\}$ *de l'ensemble des séquences infinies et observables du système telle que*

$$\forall i \in \{1, \ldots, n\}, \mathsf{mf}_i \equiv \mathsf{MF}(\Omega_i).$$

La notion de diagnosticabilité de macromodes est plus laxiste et généralise le concept de diagnosticabilité de mode. Soit $\{\mathsf{f}_i\}_{i \in \{1,\ldots,n\}}$ les modes de faute du système alors le système est diagnosticable si et seulement si l'ensemble de macromodes $M_{min} = \{\{\mathsf{ok}\}, \{\mathsf{f}_1\}, \ldots, \{\mathsf{f}_n\}\}$ est diagnosticable :

$$\mathsf{diagnosticable}_{M_{min}}(\mathcal{S}).$$

On peut aussi remarquer que si $M_{max} = \{\{\mathsf{ok} \vee \bigvee_{i\in\{1,\ldots,n\}} \mathsf{f_i}\}\}$ où les $\mathsf{f_i}$ sont les modes de faute du système alors on a *toujours* :

$$\mathsf{diagnosticable}_{M_{max}}(\mathcal{S}).$$

Dans le cas de l'exemple de la figure 5.1, il existe en fait 7 séquences observables infinies auxquelles on peut associer les macromodes suivants :

1. $\mathsf{MF}(\{o_1 o_5^\infty\}) \equiv \mathsf{ok}$
2. $\mathsf{MF}(\{o_1^\infty\}) \equiv \mathsf{f}_1 \vee \mathsf{f}_3$,
3. $\mathsf{MF}(\{o_6^\infty\}) \equiv \mathsf{f}_1$,
4. $\mathsf{MF}(\{o_2^\infty\}) \equiv \mathsf{f}_1 \vee \mathsf{f}_2$,
5. $\mathsf{MF}(\{o_3 o_2^\infty\}) \equiv \mathsf{f}_2$,
6. $\mathsf{MF}(\{o_4^\infty\}) \equiv \mathsf{f}_4$,
7. $\mathsf{MF}(\{o_3^\infty\}) \equiv \mathsf{f}_3$.

L'ensemble de ces macromodes $M = \{\mathsf{ok}, \mathsf{f}_1 \vee \mathsf{f}_3, \mathsf{f}_1, \mathsf{f}_1 \vee \mathsf{f}_2, \mathsf{f}_2, \mathsf{f}_4, \mathsf{f}_3\}$ est diagnosticable car il réalise bien une partition des signatures. Par contre l'ensemble $M_{min} = \{\mathsf{ok}, \mathsf{f}_1, \mathsf{f}_2, \mathsf{f}_3, \mathsf{f}_4\}$ ne l'est pas. Le problème est que l'on ne peut pas toujours différencier f_1 de f_2, dans ce cas, on assouplit la contrainte en disant que l'on peut reconnaître $\mathsf{f}_1 \vee \mathsf{f}_2$. Quant à M_{max} il correspond à l'ensemble $\{\mathsf{ok} \vee \mathsf{f}_1 \vee \mathsf{f}_2 \vee \mathsf{f}_3 \vee \mathsf{f}_4\}$ associé à la partition des signatures $\pi_{max} = \{\{o_1 o_5^\infty, o_1^\infty, o_6^\infty, o_2^\infty, o_3 o_2^\infty, o_4^\infty, o_3^\infty\}\}$.

Les ensembles M_{min} et M_{max} font partie d'un ensemble partiellement ordonné d'ensembles de macromodes M du plus fin (M_{min}) au moins fin (M_{max}). Si par ailleurs, pour un ensemble de macromodes M donné, la condition de diagnosticabilité est remplie alors nécessairement elle l'est pour un ensemble de macromodes M' résultant de la fusion de deux éléments de M. Cette condition de diagnosticabilité de macromodes introduit donc une notion de *niveau de diagnosticabilité* qui va être utilisée pour caractériser les systèmes autoguérissants.

Théorème 5.1 *Un système $\mathcal{S}$ est autoguérissant ssi il existe un ensemble de macromodes M tel que :*

$$\mathsf{diagnosticable}_M(\mathcal{S}) \wedge \mathsf{reparable}_M(\mathcal{S}).$$

Ce résultat démontre ainsi le lien entre la notion de réparation introduite en section 5.1.2 et le diagnostic de modes. Pour qu'un système soit autoguérissant, il suffit de déterminer un ensemble de macromodes M qui soit diagnosticable et réparable. Si les plans de réparations sont spécialisés pour chaque mode de faute

alors il faut augmenter les conditions de diagnosticabilité (afin de s'approcher d'un ensemble de macromodes de type M_{min}). Si par contre les plans de réparations sont indépendants des modes ($\mathsf{reparable}_{M_{max}}(\mathcal{S})$) alors il devient inutile de faire du diagnostic (car $\mathsf{diagnosticable}_{M_{max}}(\mathcal{S})$ est toujours vrai). Reprenant toujours l'exemple de la figure 5.1, si les plans de réparations sont spécifiques à chaque faute (et que pour le mode ok on applique le plan de réparation vide), cela revient à dire l'assertion suivante :

$$\mathsf{reparable}_{\{\mathsf{ok},\mathsf{f}_1,\mathsf{f}_2,\mathsf{f}_3,\mathsf{f}_4\}}(\mathcal{S}).$$

Dans ce cas le système est réparable mais pas autoguérissant. L'autoguérison demande à ce que l'on ait $\mathsf{diagnosticable}_{\{\mathsf{ok},\mathsf{f}_1,\mathsf{f}_2,\mathsf{f}_3,\mathsf{f}_4\}}(\mathcal{S})$ mais ce n'est pas le cas. En particulier, f_1 et f_2 ne sont pas toujours discriminables donc il n'est pas possible de décider dans ce cas quel plan de réparation doit être appliqué. Maintenant, s'il existe de plus un plan de réparation qui peut être appliqué sur f_1 ou sur f_2 et un autre qui peut être appliqué sur f_1 ou sur f_3 alors on a :

$$\mathsf{reparable}_{\{\mathsf{ok},\mathsf{f}_1\vee\mathsf{f}_3,\mathsf{f}_1,\mathsf{f}_1\vee\mathsf{f}_2,\mathsf{f}_2,\mathsf{f}_4,\mathsf{f}_3\}}(\mathcal{S})$$

et l'autoguérison du système est ainsi garantie.

5.2 Diagnostic actif sur les systèmes à événements discrets

Le diagnostic actif est un processus qui consiste à agir sur le système afin d'affiner le diagnostic courant du système. Le diagnostic actif peut être simplement la sélection de tests, la sélection de nouvelles mesures (c'est ce qu'on appelle la sélection de la prochaine mesure dans le cadre des systèmes statiques). Mais, il peut être également constitué d'actions qui modifient l'état interne du système afin de générer des observations qui raffinent le diagnostic courant. Cette section présente les contributions sur l'intégration d'un raisonnement de type diagnostic actif dans le cadre des systèmes à événements discrets.

5.2.1 Contexte des travaux

Le travail sur le diagnostic actif est issu d'une collaboration avec Elodie Chanthery, maître de conférence au sein de l'équipe DISCO et la supervision commune d'étudiants en stage de 4 ème et de 5 ème année (Julien Salvy, Nicolas Bussac). Ces contributions sont en lien avec le projet de recherche AGATA [Py et al., 2006].

5.2.2 Extension de modèle pour le diagnostic actif

Dans le cadre de la contribution sur les systèmes autoguérissants, le comportement de réparation n'est pas intégré dans le modèle explicitement (on supposait l'existence d'un plan de réparation hors modèle (voir notamment la définition 5.2)). Ici, le modèle que l'on étudie intègre les comportements d'actions sur le système. Les travaux [11, 10, 9, 12] s'appuient ainsi sur une extension des SED dans laquelle on introduit un nouveau type d'événements : l'action.

Définition 5.8 (Action) *Une action est un événement du système qui n'a lieu que si un décideur (un contrôleur) externe au système l'autorise.*

Ainsi, dans cette extension on fait la distinction entre le type d'événements, l'action (appelé événement actif) et l'événement réactif (événement non contrôlé). Cette définition d'actions conduit à plusieurs remarques. Premièrement, une action est un événement (au sens de la définition 3.1 page 58), à savoir un phénomène qui est considéré comme instantané. Ainsi la notion d'action ne modifie en rien la notion de trajectoire, la notion d'état intrinsèque au SED présentée dans le chapitre 3. Le deuxième point concerne l'observabilité de l'action. Cet événement n'a lieu sur le système que si un agent externe au système le décide, autrement dit, l'agent externe *observe* effectivement l'occurrence de l'action. Dans les travaux proposés ici, on considère que l'agent externe joue un double rôle : celui qui décide de l'action et celui qui a la charge de diagnostiquer les problèmes sur le système. Ainsi, on considérera que toute action est observable ce qui est intégré dans le masque observable obs (définition 3.6 page 60).

Hypothèse 5.1 *Toute action du SED est observable. Le masque observable obs du système est tel que si a est une action alors* $obs(a) = \{a\}$. *De plus,* $\forall e \in \Sigma \setminus \{a\}, a \notin obs(e)$.

Étant donné l'état d'un SED à un instant donné, il n'est pas forcément possible d'appliquer tous les types d'actions dans cet état. On peut même imaginer que dans des états, il s'avère impossible d'appliquer une action. Dans le cadre de cette contribution, une restriction supplémentaire sur le type de SED étudié est imposée.

Hypothèse 5.2 *Soit* τ *une trajectoire du SED* $\mathcal{S}$*, le nombre de séquences d'événements* τ' *telles que* τ' *ne contient pas d'actions et* $\tau\tau'$ *est une trajectoire de* $\mathcal{S}$ *est fini.*

L'hypothèse précédente dit que si aucune action n'est applicable dans l'état courant du système (seuls des événements réactifs peuvent survenir dans cet état), le système atteindra toujours et en un nombre fini d'événements un état dans lequel une action pourra être appliquée. Un système qui ne respecte pas cette condition est un système où il existe des trajectoires infinies sur lesquelles l'agent ne peut décider d'aucune action.

Ces cas sont rédhibitoires pour la mise en œuvre d'un diagnostic actif sur le système. En guise de conclusion de cette sous-section, la définition décrit les modèles de système considérés ici. Cette définition étend la définition 3.7 page 60 avec les actions et leurs hypothèses associées.

Définition 5.9 (Modèle du système actionnable observé) *Le modèle d'un système actionnable observé est un quadruplet* $\mathsf{DS} = (S, \Sigma_a, \Sigma_o, obs)$ *tel que :*

- S *est le modèle d'un système sur un alphabet* Σ *;*
- $\Sigma_a \subseteq \Sigma$ *est l'ensemble fini des actions sur le système;*
- Σ_o *est un ensemble d'événements observables tel que* $\Sigma_a \subseteq \Sigma_o$ *;*
- $obs^* : \Sigma^* \longrightarrow 2^{\Sigma_o^*}$ *est un masque observable séquentiel tel que*

$$\forall a \in \Sigma_a, obs(a) = \{a\}$$

et

$$\forall e \in \Sigma \setminus \{a\}, a \notin obs(e);$$

- *l'hypothèse 5.2 est respectée.*

La figure 5.2 illustre sous la forme d'un automate un modèle de système actionnable observé tel que $obs(o_1) = \{o_1\}$, $obs(o_2) = \{o_2\}$, $obs(a_1) = \{a_1\}$, $obs(a_2) = \{a_2\}$. Les actions sont les événements $\Sigma_a = \{a_1, a_2\}$. On peut en particulier noter que l'hypothèse 5.2 est respectée. Dans chaque état du système, une action peut toujours se produire après une séquence finie d'événements réactifs.

5.2.3 Diagnostic actif sur un SED

L'objectif de cette section est de caractériser formellement le problème du diagnostic actif sur un SED. Nous nous limiterons ici à des motifs d'intérêts $\mathcal{M} = F$ constitués d'événements de faute simple. L'intuition derrière la notion de diagnostic actif est la suivante : agir sur le système pour raffiner le diagnostic [Sampath et al., 1997], [Bayoudh et al., 2008], [Feldman et al., 2010b]. Nous avions déjà mentionné dans le chapitre 3 page 57 un sous problème de diagnostic actif, celui du choix de la prochaine mesure sur les systèmes statiques nommé diagnostic séquentiel ou problème de séquencement de tests, [Pattipati and Dontamsetty, 1992], [Olive et al., 2003], [Feldman et al., 2010b]. Notre démarche est différente à double titre. En premier lieu, les systèmes que l'on considère sont dynamiques et leur diagnostic peut donc changer au cours du temps. Le deuxième point est que l'on ne se limite pas à des sélections de mesures n'affectant pas l'état du système. Le diagnostic actif part du principe qu'une action régulière sur le système va conduire à une désambigüisation du diagnostic [McIlraith, 1995], [Kuhn et al., 2010].

Dans ces travaux, on considère que le système en opération est suivi par un processus de diagnostic en charge de déterminer si des événements de fautes ont eu lieu

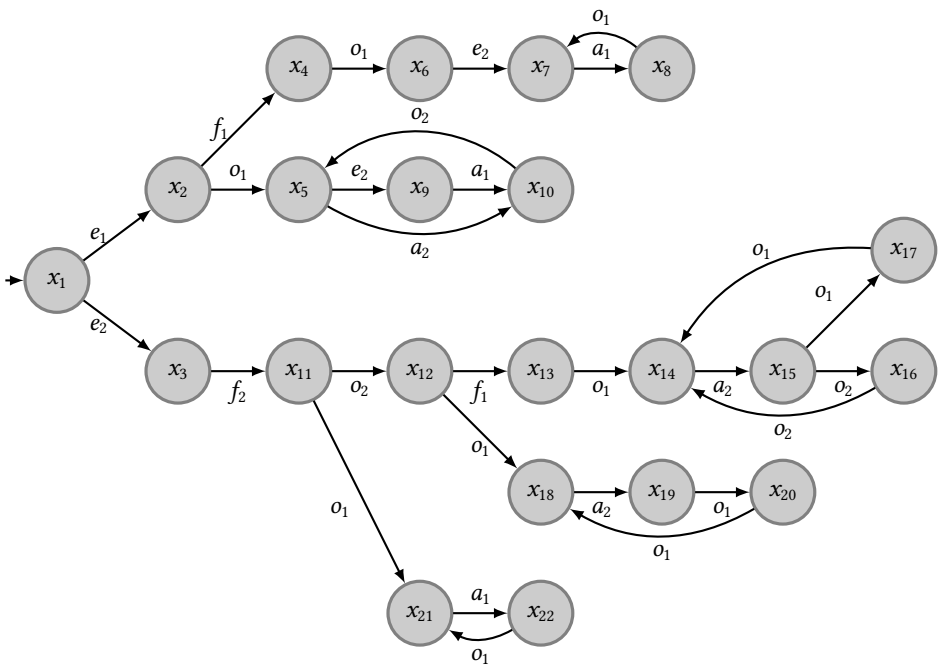


FIGURE 5.2 – Modèle de système actionnable.

ou non et si oui lesquels. À un instant donné, le système est dans un état x et a produit une séquence d'observations OBS. Le problème de diagnostic (DS, F, OBS) a pour solution un diagnostic ambigu. Le problème de diagnostic actif consiste à établir un plan conditionnel d'actions π dont l'exécution sur le système va générer de nouvelles observations OBS′ tel que le nouveau problème de diagnostic (DS, F, OBS.OBS′) a pour solution un nombre de candidats moindre que dans le problème initial (DS, F, OBS). La formalisation d'un plan conditionnel d'actions repose sur le formalisme des arbres ET-OU. Tout nœud OU a pour fils des nœuds ET et tout nœud ET a pour fils des nœuds OU. Un nœud feuille est un nœud sans fils. Un chemin dans un arbre ET-OU est une séquence de nœuds $n_1 \longrightarrow \cdots \longrightarrow n_k$ où le nœud n_{i+1} est le fils du nœud n_i pour $i \in \{1, \ldots, k-1\}$. Un chemin est maximal si n_1 est la racine de l'arbre et n_k est un nœud feuille. L'espace de recherche d'un plan de diagnostic actif peut s'exprimer à l'aide d'un tel formalisme, un plan conditionnel d'actions étant une sous-partie de cet arbre de recherche [Martelli and Montanari, 1973],[Warren, 1976],[Peot, 1992],[Pryor and Collins, 1996].

Définition 5.10 (Plan conditionnel d'actions) *Un plan conditionnel d'actions* π[DS, OBS] *sur un système actionnable* DS *ayant produit une séquence d'observations* OBS *est un arbre fini ET-OU tel que :*

- *la racine de l'arbre est un nœud ET associée à* OBS *;*
- *chaque autre nœud ET est associé à une action* $a \in \Sigma_a$ *;*

— *chaque nœud OU est associé à un langage observable* $\mathcal{O} \subseteq (\Sigma_o \setminus \Sigma_a)^*$.

Un plan conditionnel d'actions $\pi[\mathsf{DS}, \mathsf{OBS}]$ est réalisable s'il peut être appliqué sur le système $\mathcal{S}$.

Définition 5.11 (Plan réalisable) *Un plan* $\pi[\mathsf{DS}, \mathsf{OBS}]$ *est réalisable s'il existe un chemin maximal* $n_0^{ET} \longrightarrow n_0^{OU} \cdots \longrightarrow n_{k-1}^{OU} \longrightarrow n_k^{ET} \longrightarrow n_k^{OU}$ *dans* $\pi[\mathsf{DS}, \mathsf{OBS}]$ *tel qu'il existe une trajectoire* τ *dans* $\mathcal{S}$ *dont la projection observable* $\sigma = obs(\tau)$ *est telle que :*

$$\sigma \in \mathsf{OBS}.\mathcal{O}_0.a_1 \ldots \mathcal{O}_{k-1}.a_k.\mathcal{O}_k$$

où $(a_i)_{i\in\{1,\ldots,k\}}$ *désigne les actions respectivement associées aux nœuds* $(n_i^{ET})_{i\in\{1,\ldots,k\}}$ *et* $(\mathcal{O}_j)_{j\in\{0,\ldots,k\}}$ *désigne les langages observables respectivement associés aux nœuds* $(n_j^{OU})_{j\in\{0,\ldots,k\}}$.

Si un plan est réalisable, cela signifie qu'il existe au moins une *réalisation* de ce plan dans le système (à savoir au moins une trajectoire τ produisant une séquence observable σ telle que mentionnée ci-dessus).

Définition 5.12 (Diagnostic actif) *Soit* $(\mathsf{DS}, F, \mathsf{OBS})$ *un problème de diagnostic d'un système* $\mathcal{S}$*, le problème de diagnostic actif consiste à établir un plan conditionnel d'actions* $\pi[\mathsf{DS}, \mathsf{OBS}]$ *qui soit réalisable et pour lequel il existe au moins une réalisation générant une séquence observable* $\mathsf{OBS}.\mathsf{OBS}'$ *telle que la solution du problème* $(\mathsf{DS}, F, \mathsf{OBS}.\mathsf{OBS}')$ *soit strictement incluse dans celle de* $(\mathsf{DS}, F, \mathsf{OBS})$.

Le problème de diagnostic actif est donc un problème de planification d'actions sur un système avec une observabilité partielle. L'objectif du plan est défini au sens le plus large à savoir que la réalisation d'un plan de diagnostic actif se doit de minimiser l'ambiguïté du diagnostic courant en guidant le système pour qu'il produise des observations plus discriminantes. Déterminer si un plan d'actions va manifestement préciser le diagnostic, voire le déterminer précisément nécessite une analyse de diagnosticabilité du système combinée avec la synthèse du plan. Une solution possible est de créer un diagnostiqueur de SED adapté à la constitution d'un plan de diagnostic actif. C'est l'objectif de la contribution décrite dans la section suivante.

5.2.4 Diagnostiqueur actif

Le diagnostiqueur actif est une machine à états finis qui a été caractérisée dans [10]. L'objectif de cette machine est d'être capable de suivre le comportement observable du système et de fournir une information de diagnostic suffisante pour la construction éventuelle d'un plan de diagnostic actif. Ce diagnostiqueur est une adaptation du diagnostiqueur classique ([Sampath et al., 1997],[65]) pour la prise en compte de deux aspects qui sont propres au problème du diagnostic actif.

1. Le diagnostiqueur actif est généré à partir d'un modèle de SED actionnable. Dans ce cadre, le diagnostiqueur actif considère les actions comme des événements observables. En effet, il a pour objectif de suivre le comportement observable du système dans sa globalité.
2. L'information retournée par le diagnostiqueur actif est plus riche que celle du diagnostiqueur classique afin d'informer au mieux de l'intérêt de l'activation d'un plan de diagnostic actif à un instant donné. Dans un diagnostiqueur classique, on cherche à retourner pour chaque faute f si f est nécessairement présente ($f-présent$), nécessairement absente ($f-absent$) ou s'il y a ambiguïté ($f-ambigu$). Dans le cas du diagnostiqueur actif, la modalité $f-ambigu$ disparaît au profit de deux nouvelles modalités propres au diagnostic actif : $f-discriminable$ et $f-nondiscriminable$. L'intuition derrière ces deux modalités est que si le diagnostiqueur retourne $f-nondiscriminable$ on sait que l'ambiguïté ne pourra pas disparaître avec l'activation d'un plan de diagnostic actif alors que si le résultat est $f-discriminable$ il est possible de trouver un plan réalisable qui peut éventuellement lever l'ambiguïté.

Nous présentons ici la définition formelle du diagnostiqueur actif tout en sachant que sa mise en œuvre sous la forme symbolique ne pose aucune difficulté supplémentaire par rapport à la mise en œuvre symbolique du diagnostiqueur classique. Cette définition s'appuie sur une représentation du modèle de SED actionnable $\mathsf{DS} = (S, \Sigma_a, \Sigma_o, obs)$ sous la forme d'un automate $\mathcal{A} = (Q, \Sigma, T, q_0)$ (voir définition 3.18 page 64 et l'exemple de la figure 5.2). Nous rappelons également que nous limitons ici les motifs d'intérêt à de simples fautes $F = \{f_1, \ldots, f_n\} \subseteq \Sigma$.

Définition 5.13 (Diagnostiqueur actif) *Soit* $\mathsf{DS} = (S, \Sigma_a, \Sigma_o, obs)$ *un modèle de SED actionnable, soit* F *l'ensemble des événements de faute simple de* DS*, le diagnostiqueur actif* Δ *de* DS *est un automate* $(Q_\Delta, \Sigma_o, \delta_\Delta, q_{\Delta 0}, \tau_\Delta)$ *tel que :*

- Q_Δ *est un ensemble fini d'états ;*
- Σ_o *est l'alphabet du diagnostiqueur, à savoir les événements observables de* DS *incluant les actions ;*
- $\delta_\Delta : Q_\Delta \times \Sigma_o \longrightarrow Q_\Delta$ *est la fonction de transitions ;*
- $q_{\Delta 0}$ *est l'état initial ;*
- $\tau_\Delta : Q_\Delta \times F \longrightarrow Diag(F)$ *est la fonction d'étiquetage de diagnostic.*

La définition effective des états Q_Δ ainsi que de la fonction de transition δ_Δ est formellement complexe, notamment en vue d'affecter les étiquettes $f-discriminable$ et $f-nondiscriminable$ aux différents états du diagnostiqueur. Cette définition nécessite l'introduction au préalable de différentes notations. On note $chemins(Q_1, \sigma, Q_2)$ l'ensemble des chemins de transitions de $\mathcal{A}$ tels que :

1. la source du chemin est dans $Q_1 \subseteq Q$;

2. sa cible est dans $Q_2 \subseteq Q$;
3. l'application du masque observable obs sur le chemin donne exactement la séquence $\sigma \in \Sigma_o^*$;
4. la dernière transition du chemin est observable (l'application du masque observable obs sur cette transition retourne donc le dernier événement de la séquence σ).

On va noter également pour chaque état q du diagnostiqueur actif deux types d'information :

1. $\mathcal{A}(q)$ représentera un sous-ensemble d'états de Q de l'automate $\mathcal{A}$
2. $etiq(q) \in \prod_{f_i \in F} etiq(f_i)$ avec

$$etiq(f_i) \in \{f_i\text{–}absent, f_i\text{–}présent, f_i\text{–}discriminable,$$

$$f_i\text{–}nondiscriminable\}.$$

L'ensemble des états Q_Δ ainsi que la fonction de transition δ_Δ sont alors définis par induction en définissant une suite d'ensembles $Q_\Delta^0, Q_\Delta^1, \ldots$ et de fonctions $\delta^0, \delta^1, \ldots$. Ces deux suites sont définies conjointement comme suit.

1. $Q_\Delta^0 = \{q_\Delta^0\}$ contient l'état initial du diagnostiqueur. On associe également à cet état q_Δ^0 les informations suivantes $\mathcal{A}(q_\Delta^0) = \{q_0\}$ (l'état initial du diagnostiqueur est associé à l'état initial de $\mathcal{A}$) et $etiq(q_\Delta^0) = (f_1\text{–}absent, \ldots, f_n\text{–}absent)$ (dans l'état initial du diagnostiqueur, aucune faute n'a pu avoir lieu).
2. Soit $\delta^i : Q_\Delta^i \times \Sigma_o \longrightarrow Q_\Delta^{\delta_i}$, $i \geq 0$. On considère chaque état $q \in Q_\Delta^i$. Pour chaque événement observable o, on pose $\mathcal{A}(q')$ comme l'ensemble maximal (potentiellement vide) des états cibles des chemins $chemins(\mathcal{A}(q), o, \mathcal{A}(q'))$ de $\mathcal{A}$. Si $\mathcal{A}(q')$ n'est pas vide [3] notons $etiq(q', i)$ l'étiquette de f_i associée à q', alors :
   — $etiq(q', i) = f_i\text{–}absent$ si pour toute séquence observable σ telle que $chemins(\mathcal{A}(q_\Delta^0), \sigma, \mathcal{A}(q'))$ n'est pas vide, aucun de ces chemins ne contient f_i ;
   — $etiq(q', i) = f_i\text{–}présent$ si pour toute séquence observable σ telle que $chemins(\mathcal{A}(q_\Delta^0), \sigma, \mathcal{A}(q'))$ n'est pas vide, tous ces chemins contiennent f_i ;
   — $etiq(q', i) = f_i\text{–}nondiscriminable$ s'il existe au moins deux chemins dans $chemins(\mathcal{A}(q_\Delta^0), \sigma, \mathcal{A}(q'))$ dont l'un contient f_i et l'autre ne contient pas f_i et s'il n'existe pas de séquence observable $\sigma\sigma'$ telle que :
      (a) il existe un ensemble d'états $Q' \subseteq Q$ et $chemins(\mathcal{A}(q_\Delta^0), \sigma\sigma', Q')$ n'est pas vide,

3. Le cas où $\mathcal{A}(q')$ est vide caractérise un comportement non admissible. En toute rigueur pour que δ^i soit une fonction totalement définie, il faut définir $\delta^i(q, o)$ comme un état non-admissible du diagnostiqueur (état puits) [10]. Cet état puits ne sera pas considéré ici car il n'apporte rien au diagnostic.

(b) soit tous ces chemins contiennent f_i, soit aucun d'entre eux ne contient f_i.

— $etiq(q', i) = f_i\text{–}discriminable$ dans les autres cas.

3. $Q_\Delta^{i+1} = Q_\Delta^i \cup Q_\Delta^{\delta_i}$, $i \geq 0$.

Comme $\mathcal{A}$ est un automate à nombre fini d'états, il existe un nombre de pas n tel que $Q_\Delta^{n+1} = Q_\Delta^n$. L'ensemble des états du diagnostiqueur actif est finalement $Q_\Delta = Q_\Delta^n$, et la fonction de transition est $\delta_\Delta = \delta_\Delta^{n-1}$. Enfin, pour tout état q de Q_Δ et pour tout événement de faute f_i, $\tau(q, f_i) = etiq(q, i)$.

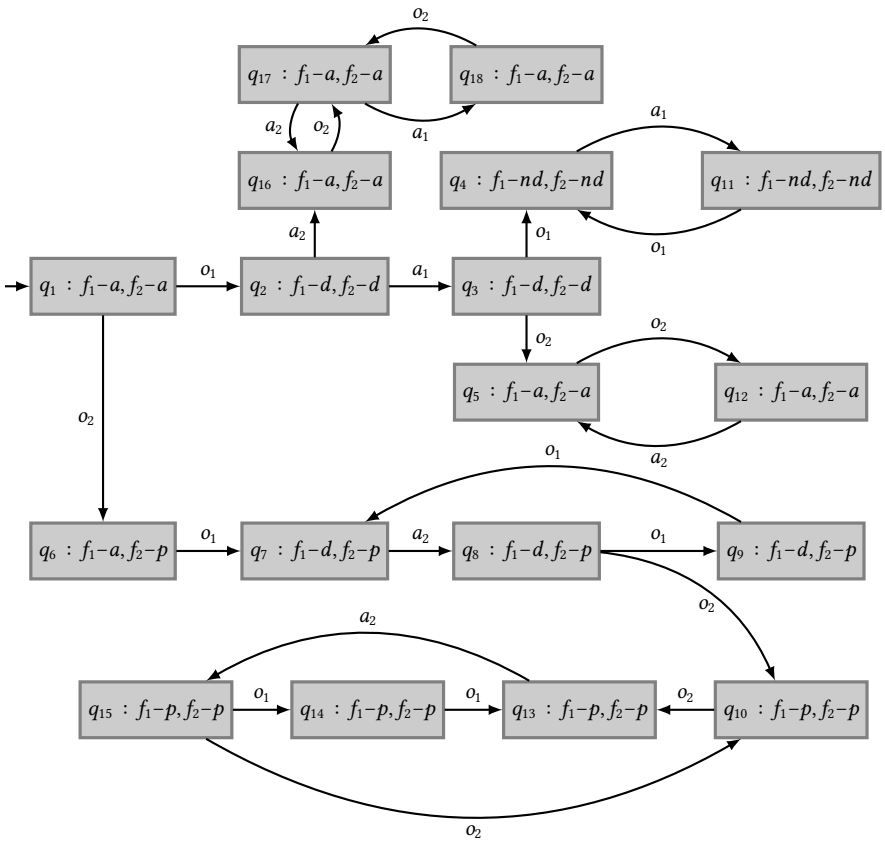


FIGURE 5.3 – Diagnostiqueur actif du modèle de système actionnable de la figure 5.2.

La figure 5.3 représente le diagnostiqueur actif du modèle actionnable de la figure 5.2. Les modalités *absent*, *présent*, *nondiscriminable* et *discriminable* sont représentées respectivement par a, p, nd, d pour des raisons de compacité graphique. Dans cet exemple, tous les cas possibles sont représentés. Les états $q_1, q_5, q_6, q_{10}, q_{12}, q_{14}, q_{15}, q_{16}, q_{17}, q_{18}$ sont équivalents aux états d'un diagnostiqueur spécialisé classique (voir la définition 3.32 page 109), ils informent sur le diagnostic courant concernant l'occurrence des événements f_1 et f_2. Les états q_7, q_8, q_9 permettent

déjà de conclure sur l'occurrence de f_2 mais pas sur celle de f_1. Dans un diagnostiqueur spécialisé, ces états seraient étiquetés f_1–$ambigu$. Ici, l'occurrence de f_1 est discriminable : il existe un plan réalisable qui permet de lever l'ambiguïté. Dans q_7, si l'on décide d'actionner a_2 [4], il existe une exécution observable du système qui mène vers un état du diagnostiqueur où on a l'assurance que f_1 a effectivement eu lieu (l'état q_{10}). Un autre cas où f_1 et f_2 sont discriminables est dans l'état q_2. Par l'action a_2, on peut immédiatement décider que f_1 et f_2 n'ont pas eu lieu. Par l'action a_1, si elle est suivie de l'observation o_1, il est garanti dans ce cas que f_1 et f_2 sont également absentes. Par contre si l'action a_1 est suivie de o_2, on rentre dans une zone de comportement non-discriminable. Dans l'état q_4, peu importe ce qu'il est possible de faire, l'ambiguïté de l'occurrence de f_1 et de f_2 ne pourra jamais être levée.

5.2.5 Utilisation du diagnostiqueur actif dans un processus de diagnostic actif

Le diagnostiqueur actif est une machine à états qui suit le comportement observable du système (il réagit aux observations du système) comme un diagnostiqueur classique. Il offre néanmoins une information plus riche pour la mise en place d'un processus de diagnostic actif.

Proposition 5.3 *Soit* $(\mathsf{DS}, F, \mathsf{OBS})$ *un problème de diagnostic sur un système* $\mathcal{S}$*, soit* $\Delta(\mathsf{OBS})$ *le diagnostic retourné par le diagnostiqueur actif* Δ *de* DS*, si toutes les fautes* $f_i \in F$ *sont* f_i–$nondiscriminable$ *dans* $\Delta(\mathsf{OBS})$ *alors le problème de diagnostic actif n'admet aucune solution.*

Cette propriété permet de décider, en fonction de $\Delta(\mathsf{OBS})$, s'il y a un quelconque intérêt de lancer un processus de diagnostic actif. La non-discriminabilité affichée dans les états du diagnostiqueur actif garantit en effet qu'un plan d'actions n'aboutira pas à une meilleure précision du diagnostic.

Proposition 5.4 *Si une faute* $f_i \in F$ *est telle que* f_i–$discriminable$ *dans* $\Delta(\mathsf{OBS})$ *alors le problème de diagnostic actif a au moins une solution par la réalisation d'un plan* $\pi[\mathsf{DS}, \mathsf{OBS}]$.

Si f_i–$discriminable$ est dans $\Delta(\mathsf{OBS})$, on sait qu'il existe un comportement observable futur qui va affiner le diagnostic sur f_i. À partir de $\Delta(\mathsf{OBS})$, on peut donc synthétiser un plan conditionnel d'actions $\pi[\mathsf{DS}, \mathsf{OBS}]$ afin de guider le système vers un tel comportement qui est réalisable par construction. Néanmoins, il faut remarquer que le comportement est conditionné par la production d'événements observables qui ne sont pas contrôlables, ainsi la réalisation souhaitée peut ne pas se produire malgré

4. Ici, c'est un cas particulier, du fait de la simplicité du modèle, seul a_2 peut avoir lieu dans q_7.

l'exécution de la bonne séquence d'actions. Si l'on revient sur l'exemple de la figure 5.3, dans l'état q_7 le plan $\pi[\mathsf{DS}, \mathsf{o_1 o_2}]$ constitué de la séquence d'actions $\mathsf{a_2 a_2}$ n'aboutit que si la réalisation observable sous-jacente est $a_2 o_1 o_1 a_2 o_2$. C'est l'observation o_2 qui est discriminante dans ce cas mais o_2 n'est pas contrôlable.

Proposition 5.5 *Tout plan* $\pi[\mathsf{DS}, \mathsf{OBS}]$ *tel que* $\pi[\mathsf{DS}, \mathsf{OBS}] = P_{\Sigma_a}(\sigma)$ *où* $\sigma \in (\Sigma_a \cup \Sigma_o)^*$ *est une séquence possible d'événements observables issue de* $\Delta(\mathsf{OBS})$ *dans* Δ *est réalisable dans* DS.

De par la proposition 5.5, l'espace de recherche d'un plan $\pi[\mathsf{DS}, \mathsf{OBS}]$ repose ainsi sur le diagnostiqueur actif et son état courant $\Delta(\mathsf{OBS})$. Tout chemin du diagnostiqueur actif issu de l'état $\Delta(\mathsf{OBS})$ est une réalisation possible d'une séquence d'actions. L'algorithme 5.1 proposé dans [12] s'appuie sur une recherche de type AO^* [Bonet and Geffner, 2000] sur le diagnostiqueur actif afin de déterminer un plan (un arbre ET-OU, voir la définition 5.10) dont la réalisation peut mener à un raffinement de diagnostic. Le choix d'une recherche AO^* est motivée par le fait que le processus de diagnostic doit fournir un plan optimal d'actions (nœuds ET) conditionné par des séquences d'observations non contrôlables (nœuds OU). La création du plan $\pi[\mathsf{DS}, \mathsf{OBS}]$ démarre par la création d'un noeud racine ET (q, OBS) associé à l'état courant q du diagnostiqueur Δ après l'observation de OBS (ligne 2). On calcule ensuite les noeuds OU successeurs de (q, OBS) (lignes 3-4), chaque noeud OU est associé à un état du diagnostiqueur q' et au langage observable (observations non contrôlées) permettant de conduire le diagnostiqueur de l'état q à l'état q'. Par l'hypothèse 5.2, ce langage est un ensemble fini de séquences finies, et le nombre de noeuds OU est fini (lignes 3-4-5 de l'algorithme 5.2). Parmi ces noeuds OU $(q', \mathcal{O})$, il y en a certains pour lesquels il est inutile de poursuivre la recherche car les étiquettes de q' sont toutes non-discriminables (lignes 6-7 de l'algorithme 5.2 et la proposition 5.3). À l'opposé, certains noeuds peuvent remplir l'objectif (lignes 9-10 de l'algorithme 5.2) : les étiquettes de q' n'expriment pas d'ambiguïté (chaque faute est soit présente, soit absente). Enfin, le dernier cas est la présence d'étiquettes *discriminables* dans q' qui garantit qu'un plan est possible en développant ce noeud (ligne 12 de l'algorithme 5.2 et la proposition 5.4). Les noeuds résolus ou à résoudre qui ont été déterminés dans calculNoeudsOU sont ensuite rajoutés dans l'arbre π en successeurs de (q, OBS) et mis dans une file (lignes 4-5 de l'algorithme 5.1). Le principe de l'itération (lignes 6-25) est d'explorer chaque nœud OU à résoudre afin de déterminer la *meilleure* action (le noeud ET) selon une heuristique qui s'appuie sur des coûts qui doivent être minimisés. Une fois un noeud ET établi on procède à l'exploration de noeuds OU qui le succèdent. Le résultat de cette itération est un arbre fini ET-OU constitué de noeuds explorés. La réduction de cet arbre menant de la racine aux noeuds résolus constituent le plan conditionnel de diagnostic actif $\pi[\mathsf{DS}, \mathsf{OBS}]$ (ligne 27).

Fonction *RecherchePlan*(DS, F, OBS)
Entrées : Soit Δ le diagnostiqueur actif de DS, F
Entrées : Soit q l'état courant de Δ après l'observation de OBS.
$\pi \longleftarrow (q, \text{OBS})$;
$noeuds \longleftarrow$ calculNoeudsOU(Δ,F,q,π);
ajouterBranches(π,(q, OBS),$noeuds$);
$file \longleftarrow noeuds$;
tant que $file \neq \emptyset$ **faire**
  $noeud = (q', \mathcal{O}) \longleftarrow premier(file)$; $supprimer(file, noeud)$;
  si $statut[noeud] =$ *à résoudre* **alors**
    $noeudsET \longleftarrow \emptyset$;
    pour tous les $a \in \Sigma_a$ **faire**
      si $\exists q'' \in \Delta, \delta_\Delta(q', a) = q''$ **alors**
        $noeudE \longleftarrow (q'', a)$;
        $heuristique[noeudE] = calculHeuristique(\Delta, q'')$;
        $noeudsET \longleftarrow noeudsET \cup \{noeudE\}$;
      fin si
    fin pour tous
    si $noeudsET = \emptyset$ **alors**
      $statut[noeud] \longleftarrow$ *non résolvable*;
    sinon
      $meilleur = (q'', a) \longleftarrow sélectionnerMeilleur(noeudsET, heuristique)$;
      ajouterBranches(π, $noeud$, $\{meilleur\}$);
      $noeuds \longleftarrow$ calculNoeudsOU(Δ,F,q'',π);
      ajouterBranches(π, $meilleur$, $noeuds$);
    fin si
  fin si
fin tq
réduirePlan(π);
Résultat : Un plan de diagnostic actif π[DS, OBS].

Algorithme 5.1 : Recherche AO^* d'un plan de diagnostic actif.

La recherche de la meilleure action s'appuie sur différents critères [Branke et al., 2004], [Smith, 2004] dont le détail est donné dans [12]. Parmi les critères à optimiser, les plus importants sont les suivants.

1. On considère que chaque action demandée a un coût associé (une dépense d'énergie, un temps d'exécution, ...) [Teung, 1994] : coût(a).
2. Chaque type de faute f_i est associé à son degré de criticité : criticité(a). Les fautes peuvent en effet avoir des effets plus ou moins critiques sur le système et sa capacité à opérer. Le fait de savoir avec certitude qu'une faute critique a eu lieu peut permettre d'effectuer des reconfigurations pour que le système puisse néanmoins travailler en mode dégradé [Chanthery et al., 2005], [Meuleau et al., 2009].

Ainsi la recherche du meilleur plan combine la minimisation des coûts sur les actions à opérer tout en maximisant la capacité à effectivement garantir la présence des fautes les plus critiques.

Fonction calculNoeudsOU(Δ,F,q,π)
Entrées : Δ le diagnostiqueur actif de DS, F
Entrées : q l'état courant de Δ.
Entrées : π plan courant partiel.
$noeudsOU \leftarrow \emptyset$;
pour tous les $q' \in \Delta, \delta^*_\Delta(q,\sigma) = q', \sigma \in (\Sigma_o \setminus \Sigma_a)^*$ **faire**
  $\mathcal{O} \leftarrow \{\sigma \in (\Sigma_o \setminus \Sigma_a)^*, \delta^*_\Delta(q_0,\sigma) = q'\}$;
  $noeud \leftarrow (q', \mathcal{O})$;
  si $objectifNonAtteignable(\Delta, \pi, q')$ **alors**
    $statut[noeud] \leftarrow$ *non résolvable*;
  sinon
    si $objectif(\Delta, q')$ **alors**
      $statut[noeud] \leftarrow$ *résolu*;
    sinon
      $statut[noeud] \leftarrow$ *à résoudre*;
    fin si
    $noeudsOU \leftarrow noeudsOU \cup \{noeud\}$;
  fin si
fin pour tous
Résultat : $noeudsOU$

Algorithme 5.2 : Recherche de nœuds OU dans le diagnostiqueur actif Δ à partir de l'état q compatible avec l'objectif.

Dans [11], on propose une architecture pour intégrer un module de diagnostic actif dans un système embarqué [Fleury et al., 1997],[Alami et al., 1998]. Cette architecture

repose sur un ordonnanceur de tâche Procosa [Barbier et al., 2006] pour la supervision de systèmes qui met en œuvre un protocole pour l'activation/désactivation du module de diagnostic actif en cohérence avec les modules de planification et d'exécution.

5.3 Diagnostic et maintenance

5.3.1 Contexte et objectif

L'activité de diagnostic consiste à déterminer entre autres la présence de dysfonctionnements au sein de systèmes industriels [Zwingelstein, 1995, Isermann and Ballé, 1997, Gertler, 1998, Isermann, 2005, Vachtsevanos et al., 2006, Ressencourt et al., 2006, Philippot et al., 2012], de systèmes automobiles [Olive et al., 2003, Soldani et al., 2007] ou encore aéronautiques [Byington et al., 2004]. Ainsi le diagnostic est une étape clé de la maintenance de ces systèmes [afn, , Kothamasu et al., 2006, Camci et al., 2007]. C'est dans le cadre du projet ARCHISTIC [20, 16, 54, 55, 52, 15] et des travaux de thèse de Pauline Ribot [53, 52, 57, 56, 54, 55, 59, 58, 60] que des travaux ont été initiés sur l'exploitation du diagnostic de défaillance dans des systèmes complexes de par la nature hétérogène de leurs composants. La contribution ici est double. Dans un premier temps, une formalisation générique du problème de diagnostic pour l'aide à la maintenance classique a été proposée. Suite à cette contribution, la formalisation a été étendue pour intégrer au raisonnement de diagnostic, un raisonnement de pronostic en vue d'assister à la maintenance prévisionnelle de systèmes hétérogènes. C'est enfin dans le cadre du projet CORALIE [27] qu'une étude plus spécifique sur l'intégration du diagnostic et du pronostic a été proposée.

5.3.2 Formalisation générique pour le diagnostic d'un système intégré en vue de sa maintenance

Un système dit *intégré* est un système construit par l'homme résultant de l'assemblage de composants de natures complètement différentes [Biswas and Manders, 2006]. L'exemple type de système intégré est l'avion : il est le résultat de l'assemblage d'éléments de nature hydraulique, pneumatique, mécanique, électronique, informatique, etc. Chaque type d'éléments a des propriétés, des dynamiques bien différentes d'un autre type et le défi de l'intégrateur est de développer un assemblage cohérent afin que tous ces éléments, *ensemble*, puissent assurer la/les *fonction(s) globale(s)* du système (à savoir pour notre cas d'étude, l'avion doit être capable de décoller, de voler, d'atterrir...). Un autre défi de l'intégrateur est qu'il doit être capable de fournir à l'utilisateur du système (dans notre cas les compagnies aériennes) un système de maintenance efficace pour l'entretien, la réparation du

système en cas de défaillance. Il se doit donc d'être en mesure de mettre en œuvre un mécanisme de surveillance et de diagnostic sur le système *dans sa globalité* sachant que les composants qui le constituent sont de nature hétérogène. C'est dans cette optique que la caractérisation logique de [60] est proposée dans les sous-sections suivantes. [5] Cette caractérisation impose un ensemble de prérequis que toute architecture de diagnostic se doit de respecter afin de garantir que les résultats de diagnostic soient logiquement cohérents entre eux. C'est notamment en analysant une architecture de diagnostic réelle vis-à-vis de cette caractérisation que l'on peut déterminer des défauts dans l'architecture de diagnostic qui peuvent conduire à des diagnostics incorrects (c'est une activité de méta-diagnostic comme celle présentée dans la section 2.4 page 38).

On considère dans notre caractérisation qu'un système intégré symbolisé par S est un système dynamique [Kuznetsov, 2004] : indépendamment de sa nature (discrète, continu, hybride) il peut être représenté en toute généralité par un triplet $(\mathsf{T_S}, \mathsf{X_S}, (f^t)_{t\in\mathsf{T_S}})$ où $\mathsf{T_S}$ est l'ensemble des temps, $\mathsf{X_S}$ est l'espace d'états et $(f^t)_{t\in\mathsf{T_S}}$ est une famille d'opérateurs d'évolution telle que :

- $f^t : \mathsf{X_S} \longrightarrow \mathsf{X_S}$ pour tout $t \in \mathsf{T_S}$;
- $f^0(x) = x$ pour tout $x \in \mathsf{X_S}$ (le système n'a pas de comportements spontanés) ;
- $f^{t+s}(x) = f^t(f^s(x))$ pour tout $x \in \mathsf{X_S}$ où les deux membres de l'équation sont effectivement définis (les règles qui définissent le comportement du système ne varient pas au cours du temps).

5.3.2.1 Caractérisation logique du contexte structurel

Le modèle structurel part du principe que le système intégré S est constitué d'un ensemble de composants qui interagissent entre eux. La structure pouvant évoluer au cours du temps, le modèle structurel du système peut être vu comme une succession de contextes structurels. On note CS un tel contexte structurel sur un espace des temps $\mathsf{T} \subseteq \mathsf{T_S}$:

$$\mathsf{ContexteStructurel(CS, T)}.$$

Le prédicat

$$\mathsf{Composant(CS, T, C, S)}$$

signifie que C est un composant du système S dans le contexte structurel CS sur l'espace des temps T. Chaque composant a sa dynamique propre caractérisée par un ensemble de variables qui vont définir son espace d'états :

5. La vision présentée ici est sous la forme logique contrairement à celle proposée dans les diverses publications sur le sujet.

$$\mathsf{Composant}(CS, T, C, S) \Longrightarrow \exists X_C \subseteq X_S, \forall x \in X_C, \mathsf{VariableEtat}(x, C, T).$$

Dire que le système est constitué d'un ensemble de composants revient à dire que chaque composant est une entité propre indépendante des autres composants. Chaque variable d'état $x \in X_S$ du système S est une variable d'état d'un composant C et d'un seul :

$$x \in X_S \equiv \exists! C, \mathsf{Composant}(CS, T, C, S) \wedge \mathsf{VariableEtat}(x, C, T).$$

Le système fonctionne dans un contexte structurel CS qui régit les différentes interactions entre les composants et les interactions entre les composants et l'environnement. Les interactions entre composants sont caractérisées par des échanges de flux entre composants, une variable x_1 d'un composant C_1 est ainsi liée à une variable x_2 d'un composant C_2 par une contrainte d'égalité :

$$\mathsf{Interaction}(CS, T, x_1, x_2) \Longrightarrow \forall t \in T, x_1(t) = x_2(t).$$

$$\mathsf{Interaction}(CS, T, x_1, x_2) \equiv \mathsf{Interaction}(CS, T, x_2, x_1).$$

Les échanges de flux peuvent s'effectuer entre n, $n \geq 2$ variables :

$$\mathsf{Interaction}(CS, T, x_1, x_2) \wedge \mathsf{Interaction}(CS, T, x_2, x_3) \Longrightarrow \mathsf{Interaction}(CS, T, x_1, x_3).$$

Le système interagit également avec son environnement, il existe ainsi un sous-ensemble de variables dans le contexte structurel qui représente ces interactions environnementales :

$$\mathsf{InteractionEnvironnementale}(CS, T, x).$$

Le contexte structurel impose une partition des variables d'états de chaque composant entre les variables qui sont en interaction, soit avec l'environnement, soit avec des variables d'un autre composant, ce sont les *variables publiques :*

$$\mathsf{Public}(CS, T, x) \equiv \mathsf{InteractionEnvironnementale}(CS, T, x)$$

$$\vee \exists x', \mathsf{Interaction}(CS, T, x, x').$$

et les variables qui n'ont pas de telles interactions et sont internes à un composant, ce sont les variables *privées* :

$$\mathsf{Prive}(CS, T, x) \equiv \neg \mathsf{Public}(CS, T, x).$$

Pour conclure cette section, on peut formaliser le contexte structurel CS de la façon suivante :

$$\mathsf{ContexteStructurel}(\mathsf{CS},\mathsf{T}) \equiv \bigwedge_{\mathsf{C}} \mathsf{Composant}(\mathsf{CS},\mathsf{C},\mathsf{T},\mathsf{S}) \wedge$$
$$\bigwedge_{x_1,x_2} \mathsf{Interaction}(\mathsf{CS},\mathsf{T},x_1,x_2) \wedge$$
$$\bigwedge_{x} \mathsf{InteractionEnvironnementale}(\mathsf{CS},\mathsf{T},x)$$

5.3.2.2 Caractérisation logique du contexte fonctionnel

Soit un contexte structurel CS il est possible désormais de caractériser un *contexte fonctionnel*. Le contexte fonctionnel établit une relation d'entrée et de sortie sur les variables publiques de chaque composant. Dans ce qui suit, on suppose que le contexte structurel CS est fixé sur un espace des temps $\mathsf{T_{CS}}$. Le contexte fonctionnel CF quant à lui est fixé sur un espace des temps $\mathsf{T_{CF}} \subseteq \mathsf{T_{CS}}$. Afin d'éviter des lourdeurs dans les prédicats suivants, les contextes structurel CS et fonctionnel CF seront toujours omis des prédicats ainsi que leur espace des temps respectifs dans cette section et les sections suivantes.

Le fait que x soit une variable de sortie du composant C dans le contexte fonctionnel CF est noté :

$$\mathsf{Sortie}(\mathsf{C},x).$$

Pour une variable d'entrée, on note :

$$\mathsf{Entree}(\mathsf{C},x).$$

Un composant dont une entrée x est en interaction avec l'environnement est une entrée du système :

$$\mathsf{EntreeSysteme}(x) \equiv \exists \mathsf{C}, \mathsf{Composant}(\mathsf{C},\mathsf{S})$$
$$\wedge \mathsf{Entree}(\mathsf{C},x) \wedge \mathsf{InteractionEnvironnementale}(x).$$

Une sortie du système est également une sortie d'un composant :

$$\mathsf{SortieSysteme}(x) \equiv \exists \mathsf{C}, \mathsf{Composant}(\mathsf{C},\mathsf{S})$$
$$\wedge \mathsf{Sortie}(\mathsf{C},x) \wedge \mathsf{InteractionEnvironnementale}(x).$$

Le lien entre le contexte fonctionnel et le contexte structurel est dû aux interactions. Toute variable en interaction est soit une variable de sortie, soit une variable d'entrée.

$$\exists \mathsf{C}, \mathsf{Composant}(\mathsf{C},\mathsf{S}) \wedge (\mathsf{Sortie}(\mathsf{C},x) \vee \mathsf{Entree}(\mathsf{C},x))$$
$$\equiv \mathsf{Public}(x)$$

Si $n, n \geq 2$ variables sont en interaction alors au moins l'une d'entre elles est une variable de sortie, ce qui est codé par ce qui suit :

$$\mathsf{Interaction}(x_1, x_2) \Longrightarrow \exists x, C \text{ tels que}$$
$$\mathsf{Interaction}(x_1, x) \wedge \mathsf{Sortie}(C, x).$$

5.3.2.3 Caractérisation logique du modèle fonctionnel

En s'appuyant sur un contexte structurel CS et un contexte fonctionnel CF, on peut définir pour chaque composant C un ensemble de prédicats signifiants que le composant C met en œuvre des fonctions élémentaires f :

$$\mathsf{FonctionElementaire}(f, C).$$

La présence du composant C dans le système intégré est liée à sa capacité à mettre en œuvre un ensemble de fonctions élémentaires F.

$$\mathsf{FonctionsElementaires}(C, F) \equiv \forall f \in F, \mathsf{FonctionElementaire}(f, C).$$

Une fonction élémentaire f met en place des conditions entre un ensemble de variables d'entrées E_f du composant et un ensemble de variables de sortie S_f du composant. Toute la difficulté de l'intégration d'un système est la mise en œuvre de fonctions globales du système à partir des fonctions élémentaires de chaque composant. Dans des contextes structurel et fonctionnel donnés, le système intégré réalise une dépendance hiérarchique entre fonctions, cette hiérarchie peut se caractériser à l'aide d'un prédicat $\mathsf{SousFonctions}(f, F)$ qui exprime le fait que la fonction f dépend directement des fonctions F. La dépendance entre f et une de ses sous-fonctions f′ est directe c'est-à-dire qu'une sous-fonction de f′ n'est pas une sous-fonction de f.

Par définition de la fonction élémentaire, elle ne dépend d'aucune autre sous-fonction :

$$\mathsf{FonctionElementaire}(f, C) \Longrightarrow \mathsf{SousFonctions}(f, \emptyset)$$

Du point de vue du diagnostic, on va chercher à déterminer si telle ou telle fonction est disponible à un instant donné, à savoir si les sorties réelles issues de cette fonction sont bien celles qui sont attendues.

Soit f une fonction élémentaire, on dit que f est disponible de la façon suivante :

$$\forall f \text{ telle que } \exists C, \mathsf{Composant}(C, S), \mathsf{FonctionElementaire}(f, C)$$

$$\mathsf{Disponible}(f, T, t) \Longleftrightarrow \exists T_f \subseteq T, t \geq T_f, \mathsf{ConditionsFonctionnelles}(f, t, T_f, E_f, S_f, C).$$

La fonction élémentaire f est disponible au temps t si la trajectoire du système qui est réalisée jusqu'au temps t remplit les conditions entre les entrées E_f et les sorties S_f.

Pour produire les valeurs de S_f au temps t, la fonction élémentaire peut nécessiter un intervalle de temps T_f.

Soit $\tau_{x_0}(0,t)$ la sous-trajectoire définie par $\forall t' \in T_{0t}, x'_t = f^{t'}(x_0)$. La fonction élémentaire f est disponible sur la trajectoire τ à l'instant t si les conditions de fonctionnement de f sont remplies sur la sous-trajectoire $\tau_{x_0}(T_f)$.

La disponibilité d'une fonction f non-élémentaire s'appuie récursivement sur la disponibilité de ses sous-fonctions $\{f_1 \dots f_n\}$.

$$\text{Disponible}(f, T, t) \equiv (t \geq T \wedge \text{SousFonctions}(f, F) \wedge$$

$$\forall f' \in F, \exists T' \subseteq T, \forall t' \in T', \text{Disponible}(f', T, t'))$$

Afin qu'une fonction f non-élémentaire soit disponible à l'instant t à la fin de l'espace des temps T, il faut que toutes les sous-fonctions le soient sur leurs temps d'utilisation T′ par la fonction f. La disponibilité des fonctions ainsi définie nécessite la disponibilité de *toutes les sous-fonctions*. Cette définition ne prend pas en compte le fait que dans les systèmes complexes et critiques comme notamment les systèmes aéronautiques, il y a des redondances architecturales qui mettent en œuvre des ensembles de fonctions redondantes. Si l'une d'entre est défaillante, une autre peut prendre le relais afin de garantir la disponibilité d'une fonction globale même en cas de défaillance d'une de ses sous-fonctions. La redondance des fonctions est intrinsèque à la nature des systèmes étudiés. Elle modifie la caractérisation de la disponibilité d'une fonction.

$$\text{Disponible}(f, T, t) \equiv (\text{SousFonctions}(f, F) \wedge F \neq \emptyset \wedge \exists N, \text{Disponibles}(F, N, t)).$$

$$\text{Disponibles}(\{f_1, \dots, f_m\}, N, t) \equiv \exists F \subseteq \{f_1, \dots, f_m\}, \|F\| = N,$$
$$\forall f \in F, \exists T' \subseteq T, \text{Disponible}(f, T', t).$$

Cette nouvelle caractérisation de la disponibilité indique que seule la disponibilité d'un sous-ensemble de sous-fonctions suffit à la disponibilité de la fonction globale. La prise en compte de la redondance de fonctions est nécessaire dans le diagnostic pour la maintenance. En effet, la décision de maintenance peut être différente si la fonction en défaillance est en redondance ou non. S'il n'y a pas de redondance, une décision de maintenance en urgence doit être effectuée. S'il y a redondance, la fonction globale peut toujours être disponible. La réparation de la fonction défaillante peut être différée (planifiée) afin d'en réduire les coûts.

5.3.2.4 Caractérisation logique du modèle comportemental

Si l'on ne s'appuie seulement que sur une caractérisation fonctionnelle du système, le diagnostic que l'on peut mener est alors un diagnostic de défaillance qui ne prend

pas encore en considération les aspects matériels qui mettent effectivement en œuvre les fonctions et en particulier les fonctions élémentaires. Comme décrit dans la section précédente, une fonction élémentaire est disponible à un instant t si des conditions fonctionnelles sont remplies entre les entrées et les sorties d'un composant lors de la réalisation du système jusqu'à cet instant t. Ces conditions fonctionnelles résultent de la dynamique interne du composant C en question qui s'appuie sur toutes les variables d'états de C et en particulier les variables privées. Pour qu'une fonction élémentaire ne soit pas disponible il y a deux cas à distinguer.

1. L'entrée E_f est correct sur le composant C, autrement dit, cette entrée a été anticipée en conception et la fonction f est supposée la prendre en compte. Les conditions fonctionnelles ne sont pas respectées à savoir que la sortie S_f n'est pas celle attendue. Dans ce cas, on peut incriminer les variables privées du composant C. L'état des variables privées constitue alors un *mode de faute*. L'indisponibilité de la fonction s'explique ainsi par un mode de faute sur le composant C. En terme de maintenance, un composant en mode de faute doit être remplacé.
2. L'entrée E_f est incorrect sur le composant C, autrement dit, cette entrée n'a pas été anticipée en conception et la fonction f n'est pas supposée la prendre en compte. Les conditions fonctionnelles ne sont pas respectées par les entrées E_f. Dans ce cas, on ne peut pas incriminer les variables privées du composant C. L'état des variables du composant constitue alors un *mode anormal*. L'indisponibilité de la fonction s'explique ainsi par une défaillance en amont du composant C. En terme de maintenance, un composant en mode anormal ne doit pas être remplacé, plus exactement son remplacement ne résoudra pas le problème.

Un mode M dans un composant C est donc défini par la disponibilité et ou l'indisponibilité de ses fonctions élémentaires.

$$\mathsf{mode}(C, M(F), t) \equiv \forall f \in F, \mathsf{FonctionElementaire}(C, f) \wedge \neg \mathsf{Disponible}(f, t) \wedge$$

$$\forall f \notin F, \mathsf{FonctionElementaire}(C, f) \wedge \mathsf{Disponible}(f, t).$$

Un mode M(F) est un mode de faute s'il existe $f \in F$ tel que sur l'espace des temps T_f sur lequel ConditionsFonctionnelles(f, t, T_f, E_f, S_f, C) n'est pas satisfait les conditions sur les entrées E_f sont respectées (elles sont dans le domaine de fonctionnement de f).

Un mode M(F) est un mode anormal s'il existe $f \in F$ tel que sur l'espace des temps T_f sur lequel ConditionsFonctionnelles(f, t, T_f, E_f, S_f, C) n'est pas satisfait les conditions sur les entrées E_f ne sont pas respectées (elles ne sont pas dans le domaine de fonctionnement de f).

Dès lors qu'un composant est dans un mode de faute, des conditions fonctionnelles ConditionsFonctionnelles(f, t, T_f, E_f, S_f, C) ne sont pas satisfaites alors que les entrées E_f sont qualifiées de *normal*. Ainsi lorsque le composant est dans un mode de faute il

est dans un domaine de fonctionnement où au moins une variable privée x n'évolue pas dans son domaine normal Dom(x).

$$\mathsf{mode}(C, M(F), t) \wedge \mathsf{modeFaute}(M(F), t) \Longrightarrow$$

$$\exists x, \mathsf{Prive}(CS, T, x), t \in T, \exists f \in F, \exists t' \in T_f, x(t') \notin \mathsf{Dom}_{ok}(x).$$

5.3.2.5 Diagnostic d'un système intégré

Dans le cadre du diagnostic pour la maintenance tel que proposé dans [16, 60], on fait l'hypothèse que le système intégré évolue dans un seul contexte structurel CS et un seul contexte fonctionnel CF, de plus, le système intégré évolue en fonction d'une entrée qui a été anticipée au moment de la conception. Ceci revient à dire que si une des fonctions du système n'est pas correcte car elle renvoie des sorties incorrectes, on fait l'hypothèse que cette défaillance s'explique par la présence d'au moins un mode de faute dans un composant. L'*état de santé* du système à un instant t est un vecteur de modes $(m_1, \ldots, m_n)$ où chaque m_i correspond au mode du composant C_i. Le diagnostic pour un système intégré peut alors être vu comme un processus qui va chercher à déterminer à partir des observations possibles quel est cet état de santé.

De par la nature hétérogène des éléments qui composent un système intégré, le problème de diagnostic ne peut être résolu par une méthode centralisée. Pour un diagnostic précis, il est nécessaire de mettre en œuvre une architecture de diagnostic où chaque constituant est une méthode de diagnostic qui est la plus adéquate pour un composant donné. En terme de maintenance, une architecture décentralisée de diagnostic [Debouk et al., 2002a, 41, Philippot et al., 2012] est la plus pertinente. Dans une telle architecture, on établit en premier lieu un diagnostic local.

Définition 5.14 (Diagnostic local) *Soit* $\mathsf{OBS}_C(t)$ *le flux d'observations issu du composant* C *jusqu'à l'instant t, le diagnostic local consiste à déterminer l'ensemble des modes*

$$\mathsf{mode}(C, M, t)$$

tel qu'il existe une trajectoire possible $\tau(t)$ *du composant* C *dont la projection observable* $obs(\tau(t))$ *contient* $\mathsf{OBS}_C(t)$ *(notée* $\tau(t) \rightsquigarrow \mathsf{OBS}_C(t)$*) qui conduit au mode* M *à l'instant t. On note :*

$$\Delta_C(t) = \bigvee_{\tau(t) \models M, \tau(t) \rightsquigarrow \mathsf{OBS}_C(t)} \mathsf{mode}(C, M, t).$$

Ensuite on fusionne les résultats en imposant les contraintes structurelles du système pour obtenir le diagnostic global.

Définition 5.15 (Diagnostic global) *Soit* $\mathsf{OBS(t)}$ *le flux d'observations issu du système* S *jusqu'à l'instant t, le diagnostic global consiste à déterminer l'ensemble des conjonctions de modes*

$$\bigwedge_{\mathsf{C,Composant(C,S)}} \mathsf{mode(C, M_C, t)}$$

tel qu'il existe une trajectoire possible $\tau(t)$ *du système* S *dont la projection observable est* $\mathsf{OBS(t)}$ *qui conduit au mode* $\mathsf{M_C}$ *à l'instant t pour le composant* C*. On note :*

$$\Delta(t) = \bigvee_{\tau(t)\models \mathsf{M_1...,M_n}, \tau(t)\rightsquigarrow \mathsf{OBS(t)}} \bigwedge_{\mathsf{C_1,...,C_n}} \mathsf{mode(C_i, M_i, t)}.$$

Le principe d'une architecture décentralisée est de mettre à profit les interactions entre les composants définis par le contexte structurel pour fusionner les diagnostics locaux afin de déterminer le diagnostic global, une telle architecture pour être correcte doit garantir l'équivalence suivante [41] :

$$\Delta(t) \equiv \bigwedge_{\mathsf{C,Composant(C,S)}} \Delta_{\mathsf{C}}(t) \wedge \mathsf{ContexteStructurel(CS)}.$$

Elle signifie que les diagnostics locaux, s'ils sont ensuite contraints par le contexte structurel sous-jacent, doivent pouvoir être raffinés pour obtenir le diagnostic global.

5.3.3 Intégration diagnostic-pronostic

La caractérisation qui a été présentée précédemment avait pour objectif de définir par un ensemble logique de prérequis le problème du diagnostic sur un système intégré en vue d'assister les décisions de maintenance classique : déterminer les composants qu'il faut remplacer parce qu'ils sont en mode de faute. Plus récemment et du fait de gros enjeux économiques sur la *disponibilité maximale* des systèmes, la maintenance dite *prévisionnelle* devient primordiale [Lebold and Thurston, 2006]. Le principe de la maintenance prévisionnelle consiste à estimer le *temps de vie résiduelle utile* d'un composant [Engel et al., 2000] afin de le remplacer avant qu'il ne soit défaillant. Afin de remplir l'objectif de la maintenance prévisionnelle, il devient alors nécessaire de résoudre des problèmes de pronostic de vieillissement [Goh et al., 2006]. L'objectif de notre contribution ici a été d'étendre la caractérisation afin d'y intégrer le problème de pronostic et de mettre en relation le problème de diagnostic et le problème de pronostic.

5.3.3.1 Caractérisation du pronostic d'un composant

Toute démarche de pronostic s'appuie sur le fait que les composants du système intégré *vieillissent* et que l'on peut déterminer la façon dont ils *vont vieillir*. Ainsi, derrière toute activité de pronostic, il y a une *loi de vieillissement* [Brotherton et al., 2000, Ghelam et al., 2006, Roemer and Byington, 2007,

Schwabacher and Goebel, 2007, Keller, 2007] Un composant est à remplacer s'il n'est pas en mesure de mettre en œuvre ses fonctions élémentaires, s'il est en mode de faute. Le vieillissement du composant suppose qu'avec le temps le composant va passer irrémédiablement en mode de faute. Or un mode de faute est caractérisé par la présence d'une variable privée qui sort de son domaine normal. En résumé, une loi de vieillissement est une fonction d'évolution d'une variable privée d'un composant, qui avec une dynamique lente, va conduire cette variable à sortir de son domaine nominal et rendre indisponible des fonctions élémentaires.

Une loi de vieillissement régie ainsi le comportement dynamique d'une variable privée x de domaine $Dom(x)$ d'un composant. Toute variable privée qui est régie par une loi de vieillissement est une *variable de dégradation*. Au même titre que pour un système dynamique, elle peut être modélisée par une famille de lois d'évolution $(f^t_{vx})_{\{t\in T\}}$ telle que $f^t_{vx} : X_C \longrightarrow Dom(x)$ où X_C est l'espace d'états du composant C dans lequel x est privé (cette famille est en fait une composante de la fonction d'évolution générale $(f^t)_{\{t\in T\}}$ définie dans la section 5.3.2).

$$\forall t \in T, x(t) = f^t_{vx}(X_C(t_0)) = f^t_{vx}(x(t_0), \dots).$$

Les variables de X_C qui influent effectivement sur x sont des *variables de stress*. Si une variable de stress est privée, elle exprime un phénomène d'usure physique du composant (rouille, poussière, fluidité,…). Elle peut également être publique (sollicitation extérieure qui vient user le composant, telle des surtensions, des températures élevées, des taux d'humidité,…).

Comme on l'a vu, la perte d'une fonctionnalité élémentaire dans un composant C est liée à des variables privées qui sortent de leur domaine normal $Dom_{ok}(x) \subseteq Dom(x)$. Ainsi, on peut définir une fonction *temps de vie restant de x* notée $tvr_x : T \longrightarrow T$ et définie par :

$$tvr_x(t) = t_d - t$$

tel que $x(t + t_d) = f^{t_d}_{vx}(X_C(t))$ et $x(t + t_d) \notin Dom_{ok}(x)$. La fonction tvr_x est une fonction qui retourne pour chaque date t le temps possible avant que le paramètre x ne sorte de son domaine normal. Cette fonction dépend de la loi de vieillissement f^t_{vx} mais aussi de l'état courant du système $X_C(t)$. Supposons à présent, qu'à l'instant t le composant soit dans un mode de faute M, sous l'hypothèse qu'il n'y a pas de changement abrupt, à quelle date le mode de faute va changer ? La perte d'une nouvelle fonction élémentaire dans le mode M est liée au passage d'une variable privée dans son domaine anormal parmi un ensemble $X^f_C \subseteq X_C$ de variables privées. Le prochain changement de mode aura lieu à la date :

$$date\_changement(M, t) = \min_{x\in X^f_C}(tvr_x(t)).$$

Si on rapporte cette caractérisation du pronostic à la littérature classique, on doit alors parler du temps de vie restant du composant C (le rul, *remaining useful life*). Supposons que le mode M du composant C à l'instant t ne soit pas un mode de faute alors :

$$rul(C, t) = date\_changement(M, t).$$

Ainsi le RUL est défini communément comme la date du premier changement de mode de faute.

5.3.3.2 Extension au pronostic de système intégré

L'objectif final de cette caractérisation est de définir le problème du pronostic sur un système intégré. Dans ce cadre, il ne peut effectivement pas se réduire à l'unique estimation du RUL de chaque composant. Du point de vue du système intégré et de sa maintenance prévisionnelle, on peut être intéressé par estimer le temps de vie utile d'une *fonction* du système. Cette analyse est motivée par la présence de redondance dans le système. En effet, la décision du remplacement d'un composant en fin de vie dont toutes les fonctions sont également mises en œuvre en redondance sur des composants neufs n'est alors pas si trivial.

Soit f une fonction, si cette fonction est élémentaire sur un composant C, alors son temps de vie restant tvr_f est lié au temps de vie restant de l'une des variables privées nécessaires à la mise en œuvre de f dont le temps de vie restant est minimal, soit x_f cette variable :

$$tvr_f(t) = tvr_{x_f}(t).$$

Si f n'est pas élémentaire alors elle a des sous-fonctions, qui plus est, certaines sont en redondance. Dans un mode du système M à l'instant t, une fonction f dispose ainsi de plusieurs ensembles F_i de fonctions disponibles pouvant assurer que la fonction f est disponible. Ainsi le temps de vie restant de f est établi par :

$$tvr_f(t) = \max_{F_i}(\min_{f' \in F_i}(tvr_{f'}(t))).$$

Le temps de vie restant de f est conditionné par le temps de vie restant maximal de l'un des sous-ensembles de sous fonctions dont la disponibilité suffit pour rendre f disponible. Au final, le système intégré réalise un ensemble de *fonctions globales* F_g, si l'une d'entre elles n'est plus disponible alors le système ne fonctionne plus correctement. C'est en partant de ce principe que l'on peut définir un RUL du système :

$$rul(t) = \min_{f \in F_g}(tvr_f(t)).$$

5.3.3.3 Apport du diagnostic au pronostic

La caractérisation du pronostic qui est proposée ici s'appuie sur l'existence de lois de vieillissement f_{vx}^{t} qui sont utilisées pour *simuler* les futurs possibles afin de déterminer les dates au plus tôt de la non-disponibilité de fonctions élémentaires liées à des variables privées qui ont quitté leur domaine de fonctionnement correct. Outre la connaissance de la loi de vieillissement qui est un défi en soi et qui est approchée par de multiples techniques (voir [59, 58] pour une caractérisation des méthodes pour approcher f_{vx}^{t} qui sont dans la plupart du temps des méthodes de type stochastique), la deuxième difficulté est liée à la connaissance de l'état courant du composant pour lancer la simulation. Plus la connaissance de l'état est précise plus la simulation des trajectoires futures et l'obtention des temps de vie restant le sera. C'est en ce point que l'activité de diagnostic apporte un élément au pronostic. En effet, diagnostiquer le système intégré revient à déterminer son état de santé courant (quels sont les modes des composants). Ces modes sont intrinsèquement déterminés par l'estimation de l'état des variables privées et donc en particulier les variables de dégradation. Ainsi, les techniques utilisées pour établir un diagnostic sur un système peuvent également être utilisé pour affiner l'estimation de l'*état de dégradation* d'un composant et donc raffiner ensuite le pronostic de ce même composant (on parle ici de diagnostic de dégradation). Dans le cadre des systèmes intégrés, l'estimation de l'état de dégradation peut également bénéficier du diagnostic global. Par exemple, on peut avoir diagnostiquer qu'un composant est en mode anormal (mais non fautif) car il subit une sollicitation anormale de ces entrées. Pour obtenir un tel diagnostic, on bénéficie des contextes structurel et fonctionnel et de l'observabilité globale du système (diagnostic décentralisé) pour estimer la valeur des entrées anormales. Ces entrées peuvent en particulier cacher des conditions de stress qui peuvent faire fortement varier la dégradation d'une variable dans le composant en mode anormal et donc leur estimation permet de raffiner le pronostic.

5.3.4 Diagnostic et pronostic d'un système dynamique différentiable par une méthode ensembliste

Afin de parachever cette section sur le diagnostic et la maintenance, cette section décrit un travail plus spécifique sur le diagnostic et son intégration avec le pronostic pour des objectifs de maintenance [Luo et al., 2008, Roychoudhury and Daigle, 2011, Daigle and Goebel, 2011, Gaudel et al., 2015]. Il résulte d'une collaboration avec la société LIEBHERR [27], l'objectif de cette étude consistant à établir si pour un composant donné d'un système aéronautique, il était possible de bénéficier d'un processus de diagnostic pour affiner le pronostic du composant cible et ainsi déterminer si ce composant pouvait entrer dans la liste des équipements sur lesquels on pourrait appliquer une stratégie de maintenance prévisionnelle. Notre étude dans ce cadre a résulté

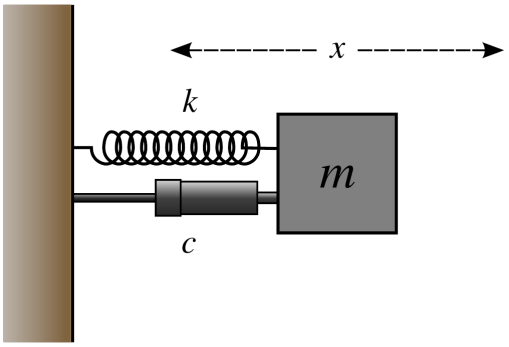


FIGURE 5.4 – Composant d'absorption de chocs.

sur le développement d'une méthode intégrant diagnostic et pronostic pour la maintenance d'un composant en s'appuyant sur le calcul ensembliste [Jaulin et al., 2001] et offre une alternative originale aux techniques statistiques [Gucik-Derigny et al., 2009, Guan et al., 2011].

5.3.4.1 Composant étudié

Pour faire le lien avec la section précédente sur la formalisation générique, le contexte des travaux décrits ici est le suivant :

- on étudie un seul composant qui est répresentable de façon homogène ;
- ce composant est dans un contexte structurel et un contexte fonctionnel fixé ;
- ce composant dispose de deux modes de fonctionnement (fonctionne/ne fonctionne pas) et d'une fonction élémentaire ;
- le composant est un système dynamique à temps continu ($T = \mathbb{R}^+$) différentiable et sa loi d'évolution est la solution d'un système d'equations différentielles ordinaires (EDO). Le contexte fonctionnel définit ses entrées comme un vecteur $\mathbf{u}(t)$ et ses sorties comme un vecteur $\mathbf{y}(t)$. La fonction élémentaire est la fonction de transfert de ce système restreinte à une enveloppe de bon fonctionnement préétablie entre $\mathbf{u}(t)$ et $\mathbf{y}(t)$.

La méthode proposée ici est détaillée sur un exemple académique mais représentatif des systèmes sur lesquels on peut l'appliquer. Il s'agit d'un absorbeur de choc constitué d'un ressort et d'un amortisseur relié à une masse [Luo et al., 2008]. La figure 5.4 illustre le composant étudié ici.

Le comportement de ce composant est régi par la seconde loi de Newton (la somme des forces est égale au produit de la masse et de l'accélération) ce qui mène au système d'équations différentielles suivant (tous les détails sont dans [68]) :

$$\begin{cases} \dot{\mathbf{x}}(t) = \begin{pmatrix} 0 & 1 \\ \frac{-k}{m} & \frac{-c}{m} \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mathbf{u}(t) \\ \mathbf{y}(t) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{x}(t) \end{cases}$$

où l'entrée $\mathbf{u}(t)$ est la force extérieure (venant de l'environnement ou d'un autre composant) appliquée à la masse m. Et la sortie est constituée de la position x de la masse ainsi que de sa vitesse $\dot{x}$ et qui constitue également le vecteur d'état $\mathbf{x}(t) = \begin{pmatrix} x(t) \\ \dot{x}(t) \end{pmatrix}$. La fonction d'évolution f^t de ce système dynamique (voir la section 5.3.2) est établie à partir des solutions de ce système d'équations et de leurs évaluations à l'instant t sur l'espace d'états X :

$$X = Dom(x) \times Dom(\dot{x}) \times Dom(\ddot{x}) \times Dom(k) \times Dom(m) \times Dom(c) \times Dom(u).$$

5.3.4.2 Diagnostic de mode de faute

Dans notre exemple, on considère que l'entrée $u(t)$ et les sorties $\mathbf{y}(t)$ sont observables mais leur observation est bruitée, le bruit étant ici géré par une méthode ensembliste qui repose sur une analyse d'intervalle. L'hypothèse du calcul ensembliste est *l'incertitude bornée* : la valeur de la variable est incertaine mais elle est, avec certitude, dans un intervalle [Moore, 1966, Jaulin and Walter, 1993, Jaulin et al., 2001]. Dans ce cas précis, on considère que si $u(t)$ est la valeur observée de u à l'instant t, la *vraie valeur* de u à cet instant est dans $[u(t) - \varepsilon_u, u(t) + \varepsilon_u]$, de même, $\mathbf{y}(t)$ est associée à une *boîte* $[\mathbf{y}](t)$ (un couple d'intervalles définissant un espace à deux dimensions sur les valeurs de $x(t) \in [x(t) - \varepsilon_x, x(t) + \varepsilon_x]$ et de $\dot{x} \in [\dot{x}(t) - \varepsilon_{\dot{x}}, \dot{x}(t) + \varepsilon_{\dot{x}}]$). Ce que l'on cherche à diagnostiquer ici est la présence d'un mode de faute. Les conditions fonctionnelles de la fonction élémentaire nous impose que les variables c et k (qui ne sont pas observables) soient dans leur domaine normal à savoir un intervalle $[c_{ok}]$ et un intervalle $[k_{ok}]$. La méthode de diagnostic consiste alors à simuler par intégration ensembliste le modèle avec l'entrée observée $[u](t)$ pour établir une estimation $[\hat{\mathbf{y}}](t)$ et la comparer avec l'observation $[\mathbf{y}](t)$. L'intégration ensembliste s'appuie sur l'outil VNODE-LP [Nedialkov, 2006]. Le résultat $[\hat{\mathbf{y}}](t)$ est une enveloppe qui couvre avec certitude de part l'hypothèse ensembliste *toutes* les trajectoires possibles du système sous l'hypothèse que les vraies valeurs de u, k, c sont dans $[u]$, $[c_{ok}]$, $[k_{ok}]$. Si l'observation $[\mathbf{y}](t)$ n'est pas *intégralement* dans $[\hat{\mathbf{y}}](t)$ alors on a la garantie que k ou c viole les conditions fonctionnelles de la fonction élémentaire qui est ainsi indisponible au temps t. Le composant est passé dans son mode de faute. Les figures 5.5 et 5.6 illustrent ces simulations et ces comparaisons. Sur la figure 5.5, $[\mathbf{y}](t)$ en rouge est totalement dans la simulation $[\hat{\mathbf{y}}](t)$ en vert : on peut préjuger que le composant n'est pas en mode de faute (mais on en est

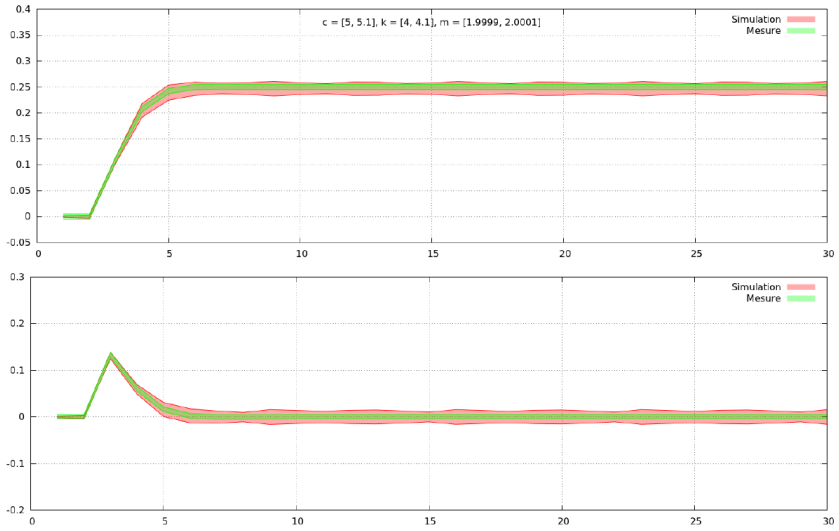


FIGURE 5.5 – Cohérence entre le modèle et les observations bruitées.

pas certain). Sur la figure 5.6, l'écart entre $[\hat{\mathbf{y}}](t)$ et $[\mathbf{y}](t)$ indique le composant est en mode de faute et le fonction élémentaire associée n'est plus disponible.

Ce type de méthode est un potentiel candidat pour la mise en œuvre du diagnostic local pour la maintenance d'un système intégré au sens de la définition 5.14 page 180.

5.3.4.3 Diagnostic de dégradation

Le diagnostic de mode consiste à vérifier que les observations sont compatibles avec les conditions fonctionnelles qui assurent de la disponibilité des fonctions élémentaires. Mais on peut aller plus loin en utilisant la même technique et procéder à un diagnostic de dégradation. Comme nous l'avons vu dans la section précédente, les conditions fonctionnelles internes garantissant la disponibilité de la fonction de l'absorbeur de choc sont que les variables c et k soient dans une boîte $[c_{ok}] \times [k_{ok}]$. Autrement dit, dans cet exemple, les variables c et k peuvent être considérées comme des variables de dégradation. Dans le modèle de comportement, on les a supposées constantes mais elles peuvent en fait varier au cours du temps du fait de l'usure du ressort k et de l'amortisseur c. Le diagnostic de dégradation consiste donc à chercher une estimation ensembliste des variables c et k au fil du temps afin de déterminer la dynamique des évolutions possibles de c et de k pour ensuite établir un pronostic.

On part du principe suivant : le diagnostic de dégradation s'effectue sur des fenêtres temporelles de petite taille comparativement à la dynamique d'évolution de c et k. Les variables c et k évoluent par des phénomènes d'usure dont la dynamique est lente, si la fenêtre de diagnostic est petite, on pourra négliger cette dynamique dans la fenêtre

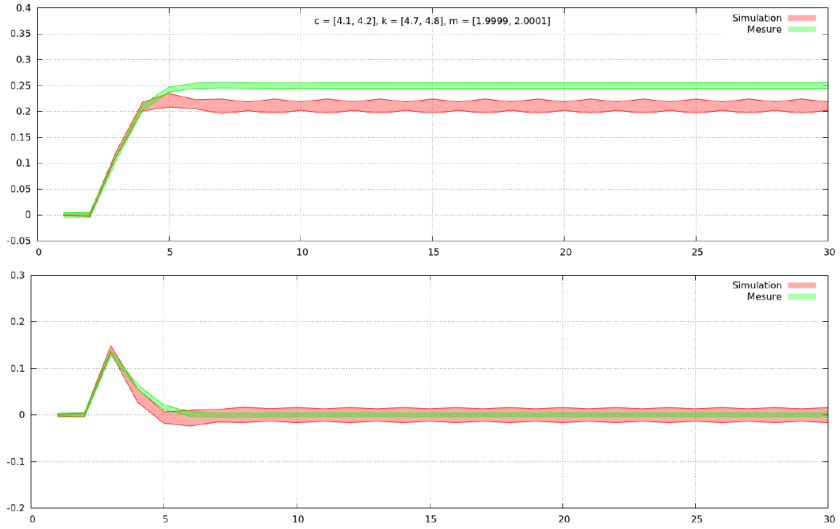


FIGURE 5.6 – Présence garantie du mode de faute.

et supposer que c et k sont constantes dans la fenêtre. On se retrouve ainsi dans la situation précédente et le modèle comportemental présenté dans la section 5.3.4.1 est valide. Pour une fenêtre temporelle donnée, on dispose ainsi des observations $[u](t)$ et $[\mathbf{y}](t)$. À l'opposé du diagnostic de mode, on ne va pas imposer que c et k soient dans une boîte unique $[c_{ok}] \times [k_{ok}]$, on va lancer une simulation pour un ensemble de boîtes qui partitionnent $[c_{ok}] \times [k_{ok}]$ afin d'avoir une estimation plus précise de c et de k sur une fenêtre temporelle donnée.

La figure 5.7 présente un tel partitionnement. La boîte qui a été partitionnée sur cette figure est $[4.51, 5.57] \times [3.85, 4.14]$ en un ensemble de boîtes dont les intervalles sont de longueurs 0.02 (exemples : en bas à gauche on a la boîte $[4.51, 4.53] \times [3.85, 3.87]$, en haut à droite on a la boîte $[5.55, 5.57] \times [4.12, 4.14]$, …). Pour chaque boîte de la partition, on simule le modèle comme pour un diagnostic de mode, trois cas se produisent :

1. les observations sont en cohérence avec la simulation sur la fenêtre temporelle étudiée (c'est le cas de la figure 5.5), tout point de la boîte représente une valeur possible pour c et k (couleur gris moyen) ;
2. les observations sont en totale incohérence avec la simulation sur la fenêtre temporelle étudiée (c'est le cas de la figure 5.6), tout point de la boîte représente une valeur impossible pour c et k (couleur sombre) ;
3. c'est le cas complémentaire : il y a un recouvrement partiel, certains points de la boîte sont possibles d'autres pas (couleur claire).

Le résultat du diagnostic de dégradation est une boîte qui assure que toutes les valeurs possibles de c et de k sont dans cette boîte (incertitude bornée). Le résultat

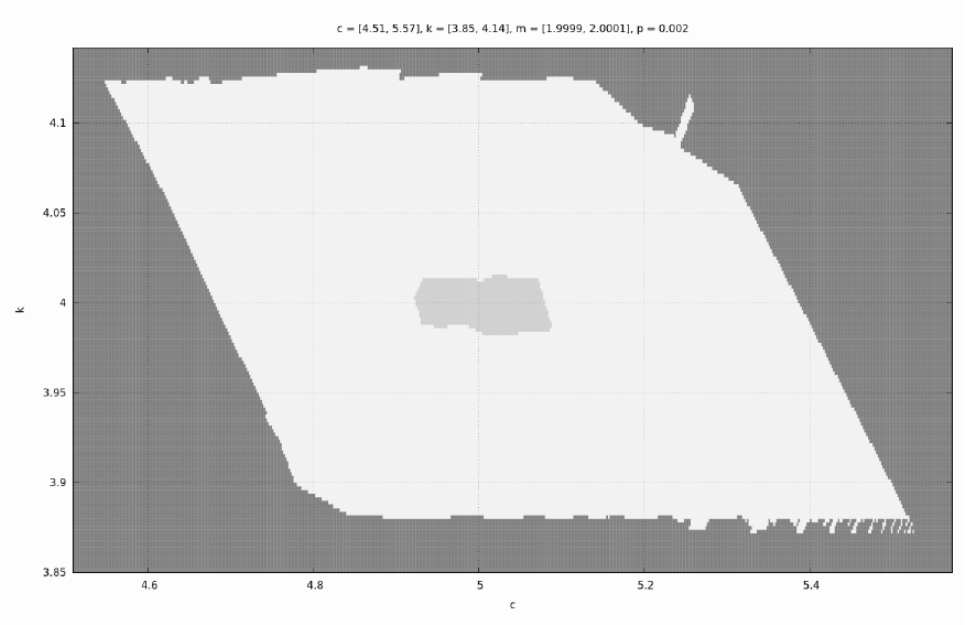


FIGURE 5.7 – Estimation de c et k. Boîte résultat : $[4.548, 5.526] \times [3.872, 4.132]$

est donc une boîte qui couvre exactement la partie claire de la figure 5.7, à savoir : $[4.548, 5.526] \times [3.872, 4.132]$.

5.3.4.4 Pronostic de vieillissement de l'amortisseur c

Le diagnostic de dégradation est utilisé pour obtenir l'estimation des variables de dégradation à l'instant t. Dans cet exemple, cette estimation a la forme d'un intervalle de valeurs possibles pour les variables de dégradation c et k à un instant donné. Cette section décrit maintenant l'étape de pronostic qui s'en suit et se focalise uniquement sur le pronostic de c (plus exactement, l'hypothèse qui est faite ici est que $tvr_k(t) >> tvr_c(t)$). Ainsi, dans ce cas précis et avec l'hypothèse précédente, le RUL du composant absorbeur est :

$$rul(absorbeur, t) = date\_changement(M_{nonFautif}, t) = tvr_f(t) = tvr_c(t).$$

Le problème de pronostic revient donc maintenant à estimer $tvr_c(t)$. La technique que l'on va utiliser est une technique de pronostic à base de modèles. Dans une telle technique, la vraie loi de vieillissement $(f^t_{vc} : X \longrightarrow Dom(c))_{t\in\mathbb{R}^+}$ (voir la section 5.3.3.1 page 181) est approchée par un modèle qui est justifié par la physique du système. Dans cet exemple, la loi de vieillissement de c est là encore décrite par une équation différentielle :

$$\dot{c} = \beta\dot{x}, \beta < 0,$$

autrement dit, le taux d'amortissement se dégrade avec la vitesse de la masse [Hutchings and Shipway, 2017]. Maintenant la difficulté est de simuler la loi de vieillissement *dans le futur*. La variable c se dégrade avec la vitesse mais cette vitesse n'est pour l'instant pas connue car c'est la vitesse future (dans ce cas précis la dégradation est régie par une famille de fonctions d'évolution définie sur l'ensemble des $\dot{x}(t'), t' > t$ possibles). Pour résoudre ce problème, on va définir des profils de fonctionnement, c'est-à-dire des schémas de comportement du système qui sont estimés comme classique sur un tel composant. On peut disposer de plusieurs classes de profils selon l'usage du composant et de son environnement [58]. Dans cet exemple, on suppose qu'il y a un seul profil moyen qui est répétitif : sur la fenêtre temporelle T_0, une force u de valeur 1 est appliquée sur la masse pendant 20 secondes à partir de $T_0 + 5$ secondes puis il y a relâchement, ce schéma peut reprendre à $T_1 > T_0 + 50$ (temps de pause entre $T_0 + 50$ et T_1). Ainsi le profil de fonctionnement que l'on considère est une séquence de cycles, chaque cycle durant 50 secondes. L'objectif maintenant est d'estimer la dégradation de c lors d'un cycle. La difficulté est que l'on ne peut plus utiliser la simulation VNODE-LP sur le modèle comportemental car sur ce modèle, on n'avait supposé que c était constant. L'estimation de c va donc s'opérer d'une manière différente mais toujours à l'aide de simulations VNODE-LP. On va tout d'abord partitionner le domaine possible de c en un ensemble de sous-intervalles $[c_1], \ldots, [c_n]$. Pour chaque $[c_i]$, on procède de la façon suivante.

1. Le cycle du profil est échantillonné avec un pas de temps δ, on dispose ainsi de N_δ sections de cycle $\mathbf{u}(T_0, T_0 + \delta)$, $\mathbf{u}(T_0 + \delta, T_0 + 2\delta), \ldots$.
2. À partir du premier échantillon $\mathbf{u}(T_0, T_0 + \delta)$ et des conditions initiales de l'état $\begin{pmatrix}[x_{0\delta}]\\ [\dot{x}_{0\delta}]\end{pmatrix}$, on simule le modèle comportemental classique en posant que $c \in [c_i]$. On obtient une estimation de l'état à $T_0 + \delta$ qui est $\begin{pmatrix}[x_{1\delta}]\\ [\dot{x}_{1\delta}]\end{pmatrix}$.
3. L'estimation $[\dot{x}_{1\delta}]$ est alors injectée dans la loi de vieillissement afin d'obtenir une nouvelle estimation de $[c_i]$: $[c_i]^{1\delta}$.
4. On réitère les opérations 2 et 3 à partir des conditions $c \in [c_i]^{1\delta}$ et d'état initial $\begin{pmatrix}[x_{1\delta}]\\ [\dot{x}_{1\delta}]\end{pmatrix}$ sur la fenêtre $\mathbf{u}(T_0 + \delta, T_0 + 2\delta)$ pour obtenir $[c_i]^{2\delta}$ et $\begin{pmatrix}[x_{2\delta}]\\ [\dot{x}_{2\delta}]\end{pmatrix}$ et ainsi de suite jusqu'à la dernière section afin d'obtenir $[c_i]^{N_\delta\delta}$.

Le résultat de cette procédure itérative est que pour chaque intervalle $[c_i]$, on va estimer que sa dégradation sur un cycle est $[c_i]^{N_\delta\delta}$. Ceci peut se résumer dans un tableau comme celui de la table 5.1. Dans cet exemple, on considère que le domaine de c est l'intervalle $[0, 10]$ qui a été partitionné en dix sections.

$[c_i]$	$[c_i]^{N_\delta \delta}$
$[9, 10]$	$[8.917, 9.977]$
$[8, 9]$	$[7.911, 8.978]$
$[7, 8]$	$[6.898, 7.979]$
$[6, 7]$	$[5.814, 6.982]$
$[5, 6]$	$[4.859, 5.979]$
$[4, 5]$	$[3.874, 4.977]$
$[3, 4]$	$[2.863, 3.973]$
$[2, 3]$	$[1.721, 2.97]$
$[1, 2]$	$[0, 1.98]$
$[0, 1]$	$[0, 0.9755]$

TABLE 5.1 – Estimation de la dégradation sur un cycle de c sur le domaine $Dom(c) = [0, 10]$.

Cette étape de création de la table peut être vue comme une phase de précompilation du modèle de vieillissement qui est établie hors-ligne. Le pronostic en-ligne consiste à exploiter le résultat du diagnostic de dégradation à l'instant t et cette table. Le pronostic procède de la façon suivante. On part du principe que le composant va subir dans le futur une séquence de cycles préétablis. À partir de l'instant t, on va simuler à l'aide de la table la dégradation de c. Reprenant l'exemple de la table 5.1, on pose que les conditions fonctionnelles associées à la fonction élémentaire de l'absorbeur sont $[c_{ok}] = [2, 10]$. Le diagnostic de dégradation à l'instant t informe que $c \in [4.548, 5.526]$ (fin de la section 5.3.4.3). On va procéder par interpolation linéaire de la table, c'est-à-dire que si $[c_i] = [a, b]$ et $[c_i]^{N_\delta \delta} = [a', b']$ et si c est une valeur de $[a, b]$ alors au prochain cycle cette valeur sera $c' = b' - \frac{(b-c)\times(b'-a')}{b-a}$. Autrement dit, selon la table 5.1, si $c \in [4.548, 5.526]$ alors au bout d'un cycle $c \in [4.4787, 5.4481]$. En propageant de cycle en cycle, on va finalement aboutir dans ce cas à des estimations de c où $c < 2$ et viole ainsi les conditions fonctionnelles. Dans cet exemple, cela se produit au plus tôt à $N_{min} = 30$ cycles (à partir de 30 cycles c est estimé appartenir à un intervalle contenant la valeur limite 2 donc il existe une possibilité que $c < 2$). Cela se produit au plus tard à $N_{max} = 44$ cycles (toutes les estimations de c sont inférieures à 2 à partir de 44 cycles). Le résultat ainsi obtenue par cette méthode est un intervalle de nombre de cycles. Le pronostic estime que le composant deviendra défaillant dans $[N_{min}, N_{max}] = [30, 44]$ cycles.

Soit maintenant $[t_{N_{min}}] = [\underline{t_{N_{min}}}, \overline{t_{N_{min}}}]$, une estimation de temps pour une séquence

de N_{min} cycles (temps de pause inclus) et $[t_{N_{max}}] = [\underline{t_{N_{max}}}, \overline{t_{N_{max}}}]$ la même chose pour N_{max} cycles alors on a finalement une estimation de $\overline{tvr_c(t)}$:

$$tvr_c(t) \in [\underline{t_{N_{min}}}, \overline{t_{N_{max}}}].$$

5.4 Résumé

L'objectif de ce chapitre a été de synthétiser toutes les contributions où le raisonnement diagnostique assiste la décision. Le premier type de décision qui est nourri par un diagnostic préalable est bien évidemment la décision de réparation. La caractérisation des systèmes autoguérissants met en évidence le lien entre la capacité de diagnostic et la capacité de réparation automatisée. Sa mise en œuvre exploite directement les contributions sur l'analyse de diagnosticabilité présentées dans la section 3.3 du chapitre 3. Ce premier travail est une étape vers l'amélioration de l'autonomie des systèmes en général. Le deuxième thème abordé ici est aussi lié à l'autonomie des systèmes. Afin d'affiner un diagnostic trop ambigu en ligne, l'objectif a été de mettre en place un processus de diagnostic actif afin de générer à la volée un plan pour affiner le diagnostic dans le système actionnable et isoler notamment les fautes critiques qui nécessitent des reconfigurations. Ce processus de diagnostic actif bénéficie d'une analyse préalable de diagnosticabilité active qui, là encore, étend les analyses décrites dans la section 3.3. Le troisième thème abordé est le diagnostic pour l'aide à la maintenance dans les systèmes intégrés. L'enjeu de ce travail a été de définir un cadre générique (un ensemble de prérequis) à partir duquel il est possible de mettre en œuvre une architecture de diagnostic décentralisée qui génère des résultats *logiquement corrects* pour assister la maintenance classique. Cette description est générique et peut être employée pour mettre en œuvre des architectures réalistes intégrant non seulement des techniques de diagnostic présentées dans ce document mais également des techniques de diagnostic de systèmes continus/hybrides qui ne font pas l'objet de ce document. Enfin, cette architecture a été étendue par une caractérisation du problème de pronostic et son lien avec le diagnostic en vue d'améliorer la maintenance prévisionnelle. Un cas d'étude spécifique, fondé sur des méthodes d'estimations ensemblistes, montre en particulier ce lien entre diagnostic et pronostic.

Chapitre 6

Quelques aspects logiciels

Les travaux qui ont été présentés dans les chapitres précédents ont donné lieu à des réalisations logicielles. Ce chapitre a pour objectif de présenter un bref état des outils développés et maintenus ou toujours en cours de développement. Je considère dans ma démarche scientifique que le développement logiciel est essentiel à plusieurs titres. Le premier bien évidemment est la production de résultats expérimentaux, mais j'estime finalement que cela n'est pas le plus important. Le deuxième intérêt est que le développement logiciel impose une rigueur supplémentaire à la théorie développée car l'on doit prendre en compte les détails techniques d'une mise en œuvre ce qui consolide les résultats avancés. Enfin, et surtout, un logiciel ne résulte pas d'une seule publication mais constitue un tout, une mise en cohérence des diverses contributions au cours du temps.

6.1 Dito : outil de diagnostic logique

L'outil Dito (*DIagnosis TOols*) est un outil développé en Java [38]. Il met en œuvre les algorithmes présentés dans les sections 2.1, 2.2, 2.4 du chapitre 2. La description générale de cet outil est en particulier présentée dans la section 2.2. Cet outil peut être utilisé de deux façons différentes, soit en ligne de commande pour résoudre des problèmes de diagnostic à base de modèle ou de meta-diagnostic [6] à partir d'une description logique du problème, soit à l'aide d'une interface graphique qui présente deux démonstrations de problèmes totalement différents mais qui utilisent pourtant le même moteur de diagnostic pour leur résolution. L'objectif de Dito est de mettre en place un langage logique de représentation d'un problème de diagnostic statique et de développer un algorithme de diagnostic qui soit indépendant de l'outil qui sert à vérifier la satisfaisabilité des problèmes posés. L'objectif final est de pouvoir ainsi bénéficier d'un jeu de solveurs possibles (CSP, SAT, SMT,...) et de profiter des avantages de l'un sur l'autre en fonction du problème de diagnostic donné. Pour le moment, Dito

ne fonctionne qu'avec un solveur CSP (Choco [Team, 2008]). Le logiciel DITO résulte du développement de MEDITO [6] par Nuno Belard lors de sa thèse et de son extension par mes soins.

6.1.1 Mode non graphique

Dans le monde non graphique de DITO, le principe est de modéliser dans un fichier le problème de diagnostic souhaité. Par exemple, le fichier de description du problème présenté sur le circuit de Davis (voir la section 2.1 et en particulier la figure 2.1 page 19) est donné sur la figure 6.1. Cette description est la simple transcription du problème en formules de logique du premier ordre. En utilisant Dito sur ce fichier, il retourne le diagnostic suivant (ensemble de diagnostics noyaux) :

```
Ab(A2), Ab(M3), Ab(M2) Ab(A1), Ab(M1) Ab(M2).
```

```
components:
Comp(A1); Comp(A2); Comp(M1); Comp(M2);  Comp(M3);

observations:
{ A=1 }; { B=2 }; { C=3 }; { D=4 }; { E=5 }; { F=11 }; { G=22 };


description:
(not Ab(M1)) implies { X = A * C };
(not Ab(M2)) implies { Y = B * D };
(not Ab(M3)) implies { Z = E * C };
(not Ab(A1)) implies { F = X + Y };
(not Ab(A2)) implies { G = Y + Z };
```

FIGURE 6.1 – Fichier de description du problème de Davis dans Dito.

C'est également à l'aide de DITO que les problèmes de méta-diagnostic présentés dans la section 2.4.6 sont résolus (voir la figure 2.5 pour un exemple).

6.1.2 Mode démonstration

DITO dispose également d'un mode graphique de démonstration. L'objectif de ces démonstrations est de mettre en évidence la généralité du problème traité en montrant deux problèmes qui n'ont strictement rien à voir mais qui font pourtant appel au même raisonnement, au même algorithme de diagnostic (voir chapitre 2, section 2.2 page 23).

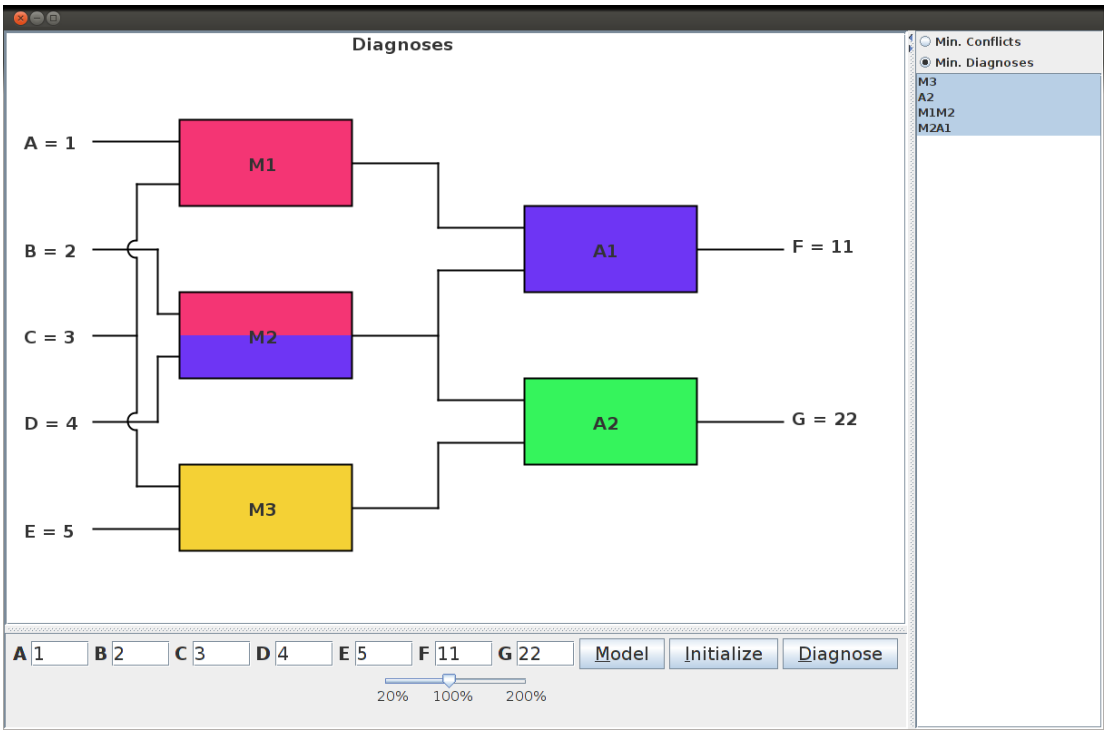


FIGURE 6.2 – Démonstration du circuit de Davis dans DITO.

La première démonstration proposée est le problème de diagnostic classique sur le circuit de Davis. La figure 6.2 représente l'interface graphique qui présente le résultat du problème posé dans la section 6.1.1 ci-dessus. Chaque couleur indique un diagnostic minimal candidat (il y a 4 candidats dont 2 sont des fautes multiples).

La deuxième démonstration présente un problème de diagnostic sur le jeu du Sudoku (voir la figure 6.3). Dans ce contexte, DITO permet de dire au joueur si au cours du jeu, il se trompe et si oui où il se trompe. Chaque configuration de jeu est en effet un problème de diagnostic en soit. Intuitivement, les composants du système sont les cellules de la grille. La fonction de chaque cellule est de fournir un chiffre qui satisfait les contraintes de son voisinage, si le chiffre n'est pas compatible alors la cellule peut être suspectée comme étant défaillante. Les observations du problème sont les chiffres qui ont déjà été sélectionnés par le joueur. Sur la figure 6.3, le joueur a choisi un chiffre pour 5 cellules, DITO retourne un diagnostic candidat de cardinalité minimal qui indique que 3 de ces cellules sont erronées.

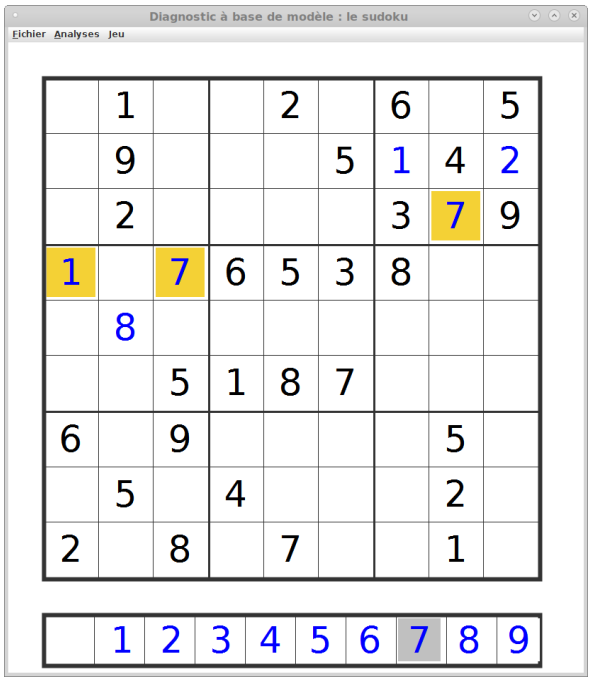


FIGURE 6.3 – Sudoku : 3 choix erronés sur cette grille.

6.2 Diades : diagnostic de systèmes à événements discrets

DIADES est un ensemble de bibliothèques écrites en C++ qui met en œuvre les techniques de diagnostic des systèmes à événements discrets et de diagnosticabilité qui ont été présentés dans ce mémoire. Chaque bibliothèque met en œuvre une des spécificités du travail exposé. J'ai réalisé la plupart des développements et assuré l'intégration du développement de mes collaborateurs (Dimitry Kamenetsky pour la partie précision, Anika Schumann pour la partie diagnostic symbolique, Nicolas Bussac pour l'algorithme de planification du diagnostic actif).

La structure de données de base de DIADES est une structure de graphe, elle a été développée dans l'objectif de simplifier les différentes mises en œuvre de problème de diagnostic qui ont été étudiés au cours de la période. C'est à partir de ces structures de graphe que DIADES met en œuvre une structure d'automate de type énumérative, une structure de réseau de Petri, ainsi que la mise en œuvre des topologies de connexions entre composants d'un système à événements discrets.

La première structure de données que DIADES met à disposition est une structure de systèmes à événements discrets définis à l'aide de composants d'automates synchronisés. À partir d'un ensemble de fichiers textuels représentants des composants et

d'un fichier définissant de façon générique une loi de synchronisation entre les composants DIADES est en mesure d'appliquer le spectre d'algorithmes (formes énumératives) présenté dans le chapitre précédent sur le modèle compositionnel défini par ses fichiers. La bibliothèque d'automates de DIADES a été développée dans le but de faciliter les opérations de projection et de synchronisation entre automates qui sont nécessaires pour la mise en place de techniques de diagnostic et d'analyse de diagnosticabilité. Par conséquent, les opérations de synchronisation et de projection mise en œuvre dans DIADES sont totalement génériques, paramétrés par le type des événements considérés. Par exemple, pour appliquer une méthode de diagnostic global, on doit dans un premier temps synchroniser les événements interactifs des automates du système, alors que pour une analyse de diagnosticabilité globale, on est souvent amené à synchroniser des automates sur leurs événements observables (produit jumelé). Dans l'analyse décentralisée présentée dans la section 3.3.3 par exemple, les synchronisations sur les observables et les interactions entre composants se font en alternance. Dans DIADES, cette opération est la même dans les deux cas, seuls les paramètres changent. Il en est de même pour les projections qui sont des opérations extrêmement utilisées dans les divers travaux que j'ai développés au cours de la période (calcul de traces observables, projection sur les événements interactifs, etc.). DIADES non seulement apporte une vue du système en tant que machines à état, mais également apporte un vue de type langage (langage régulier, automate à états accepteurs). Ainsi, DIADES est capable de déterminiser tout type d'automates, et de calculer la représentation minimale du langage régulier qu'il génère (algorithmes de déterminisation et de minimisation). Bien que DIADES soit en mesure de travailler sur des automates non déterministes et non minimaux, dans bien des situations (notamment celle de la section 3.4 au sujet de la précision), il s'est avéré plus intéressant d'un point de vue performances de travailler avec des automates minimaux déterministes, car ils correspondent à une représentation canonique des comportements diagnostiqués. À l'aide de cette bibliothèque, DIADES met également à disposition un générateur aléatoire de systèmes à événements discrets, ainsi qu'un générateur aléatoire à événement discret dont on a la garantie par construction qu'il est diagnosticable [39]. DIADES met également en œuvre des indicateurs de performance afin de déterminer la qualité du diagnostic et le temps de calcul en comparant dans la mesure du possible les différents algorithmes proposés [37].

DIADES dispose également d'une bibliothèque qui met en œuvre le spectre de diagnostic en version dite symbolique (voir la section 3.2.3). L'ensemble des algorithmes mis en œuvre repose sur une bibliothèque de diagramme de décision binaire, notre choix s'est porté sur la bibliothèque CUDD [Somenzi, 1998]. C'est à partir de cette mise en œuvre et de la mise en œuvre précédente que l'on a pu établir une étude comparative des approches dites énumératives et avec celles qui sont symboliques [64, 66].

DIADES dispose également d'une bibliothèque de réseau de Petri qui s'appuie sur

les mêmes structures que les automates à savoir des graphes, en vue de pouvoir notamment traduire les chroniques discutées dans la section 4.3.3.1 en réseau de Petri temporel en vue de travailler sur ce modèle à l'aide de l'outil TINA [Berthomieu et al., 2004]. Cette outil a notamment été utilisé pour une démonstration lors de la revue finale du projet européen WS-DIAMOND [18, 49]. C'est aussi à partir de cette bibliothèque que l'on peut construire automatiquement le problème de la vérification de diagnosticabilité qui a été défini dans la section 3.3.5. DIADES propose ainsi une interface complète avec TINA en vue de l'exploiter pour la vérification de modèles demandé dans le contexte de la diagnosticabilité et du diagnostic.

DIADES met également en œuvre les algorithmes de diagnostic actif qui ont été développés en section 5.2. Ces algorithmes s'appuient sur la notion de traces et résulte de la projection des automates du modèle sous-jacent. L'algorithme de planification de type AO* est également mis en œuvre dans cet outil.

Un des objectifs de DIADES est d'intégrer dans la même architecture logicielle des outils utiles au diagnostic d'un large spectre de systèmes à événements discrets. Dans cette optique, DIADES met également en œuvre une interface avec le langage de spécification ALTARICA [Point and Rauzy, 1999]. A l'heure actuelle, DIADES n'est en mesure de gérer que les problèmes définis dans ALTARICA qui sont purement à événements discrets (pas de variables, voir les perspectives de la section 7.2.1.4). La mise en œuvre de l'extension sur les automates à contraintes qui sont propres à ALTARICA est en cours [40].

L'outil DIADES enfin est utilisé dans le cadre des systèmes hybrides. Au sein de l'équipe DISCO, des études sur le diagnostic des systèmes hybrides ont consisté à discrétiser le problème pour revenir à un problème sur des systèmes à événements discrets. Cette étude a mené à la mise en œuvre de l'outil de diagnostic hybride HYDIAG et de son extension au diagnostic actif [13]. Cet outil discrétise le problème dans un premier temps et génère un automate qui est ensuite analysé par DIADES qui retourne un diagnostiqueur pour la résolution du diagnostic de systèmes hybrides par HYDIAG. Dans la partie réservée au diagnostic actif de HYDIAG, c'est également à l'aide du diagnostic actif développé dans DIADES que le diagnostic actif du système hybride est mis en œuvre.

6.3 Autres contributions logicielles

Cette section liste diverses autres contributions logicielles.

6.3.1 Plateforme logicielle TIPADIAG

La plateforme logicielle TIPADIAG (*Time Patterns for Diagnosis*) a pour objectif de rassembler les diverses algorithmes de l'équipe DISCO autour du formalisme

des chroniques (voir les sections 4.1, 4.2, 4.3). Cette plateforme offre une structure de données pour les chroniques et permet de lancer les analyses de qualités sur elles (voir la section 4.2.3 page 128). Elle offre également le moyen de traduire les chroniques en réseau de Petri temporel [23, 31] [1] (voir notamment la section 4.3.3.1 page 4.3.3.1) pour l'analyse de leur exclusivité par model-checking à l'aide de l'outil Tina[Berthomieu et al., 2004]. L'objectif final de Tipadiag est de mettre en cohérence tous les outils de modélisations, d'analyses [49, 23, 31], d'acquisitions automatiques [Subias et al., 2014] et de reconnaissance des chroniques pour le diagnostic temporel de système. Tipadiag est le résultat d'un développement collaboratif entre un ingénieur de recherche du LAAS (David Gauchard) et moi-même.

6.3.2 Plateforme logicielle Maxpluslin

Le développement de la plateforme logicielle Maxpluslin (*MAXPLUS LINear system*) a débuté récemment en vue de mettre en œuvre les indicateurs définis dans la section 4.4 [61, 30]. Le développement de cette plateforme a démarré au cours du stage de Claire Paya sous ma direction et celle d'Euriell Le Corronc. L'objectif est de décrire de façon ergonomique les problèmes de diagnostic à traiter (mise en place des Graphes d'Événements Temporisés, mise en place des scénarios observables), de simuler de tels systèmes et de lancer des problèmes de diagnostic. La plateforme Maxpluslin exploite la bibliothèque `libminmaxgd` [Cottenceau et al., 2000] afin de transcrire le problème sous la forme d'un système d'équations $(max, +)$ (voir les précisions dans la section 4.4)

6.3.3 Plateforme logicielle Prognospice

La plateforme logicielle Prognospice correspond à l'outil qui a été réalisé et livré aux partenaires du projet EPICE [27, 68]. Cet outil est celui qui a servi à produire les résultats de la section 5.3.4 page 184. L'une des originalités de cet outil est qu'il propose un langage de modélisation à partir duquel il est possible de modéliser en un seul fichier le problème de diagnostic de dégradation (estimation d'états d'un modèle de comportement) et le problème de pronostic (prédiction à l'aide d'un modèle de vieillissement). Le fichier de description pour le problème de diagnostic et du pronostic du ressort qui a été décrit dans la section 5.3.4 est entièrement présenté sur la figure 6.4. Pour différentier le modèle comportement nominal du modèle de vieillissement, des attributs sont utilisés (`nominal`,`damaging`,`nstate`,`dstate`,...). Il est à noter qu'un paramètre (par exemple `c`) est considéré comme une variable normale dans le modèle de comportement (`nominal`) et comme une variable d'état dans le modèle de vieillissement (`dstate`). Le modèle nominal est régi par les équations de type `nominal` et le modèle de vieillissement est régi par les équations de type `damaging`.

1. Cette fonctionnalité a été importée de Diades.

```
constants:
m = 50;
alpha = 0.2;

variables:
k:= 3.85 : real : nominal ;
c:= 3.0 : real : nominal, dstate;
x:= 0.9 : real : nstate;
v:= 0.0 : real : nstate, dinput, nobservable;
t:= 0.0 : real ;
u:= 0.0 : real : ninput, nobservable;
ratio : real : nominal;
pos: real : noutput, nobservable;
equations:
ratio = if (u < 0.5) then ( 0 ) else ( u / m ) endif : nominal,
input ;
pos = x : nominal, output;
dx/dt = v : nominal, state;
dv/dt = (( (-1 * k / m) * x) - ( (c / m) * v )) + ratio : nominal,
state;
dc/dt = alpha * abs(v) : damaging, state;
```

FIGURE 6.4 – Fichier de description du problème diagnostic/pronostic du ressort.

À partir de cette simple description et des fichiers de données correspondant à la mesure de certaines variables (entrées et/ou variables observables), PROGNOSPICE est en mesure de simuler le système, d'effectuer un diagnostic de dégradation (estimation d'états ensembliste, voir la section 5.3.4.3) et d'effectuer un pronostic sur la durée de vie résiduelle du ressort (section 5.3.4.4). L'outil `Prognospice` s'appuie sur l'intégrateur ensembliste VNODELP [Nedialkov, 2006], il est le résultat d'un développement collaboratif entre un ingénieur de recherche (Renaud Pons) et moi-même.

Chapitre 7

Bilan et perspectives

7.1 Synthèse des travaux de recherche

Tous les travaux qui ont été décrits dans ce mémoire servent le même objectif que je reprends ici du chapitre 1 :

la synthèse informatique d'un agent ou d'un ensemble d'agents ayant des capacités cognitives suffisantes pour surveiller, détecter et diagnostiquer des situations dans leur environnement.

Afin d'apporter des réponses et de nourrir cet objectif, plusieurs jeux de travaux portant sur des thématiques complémentaires ont été abordés.

7.1.1 Spectre de méthodes de diagnostic automatique

Un premier jeu de travaux effectués ont essentiellement portés sur la définition d'algorithmes de diagnostic pour différent types de systèmes en jouant avec plusieurs dimensions pour essayer d'en capturer le meilleur pour le diagnostic. Cette panoplie d'algorithmes de diagnostic portent sur les systèmes logiques statiques avec une approche de type CSP (section 2.2 page 23), sur les systèmes à événements discrets avec une approche de type BDD (section 3.2 page 64) ou sur les systèmes à événements temporisés en s'appuyant sur des approches de type graphe d'événements temporisés (section 4.4 page 143). Chaque ajout d'une dimension (que cela soit le passage des systèmes statiques aux systèmes dynamiques événementiels ou encore celui vers des systèmes dynamiques événementiels temporisés) apporte son degré de complexité mais en contrepartie il donne l'espoir d'une meilleure connaissance du système et surtout d'une meilleure observation de celui-ci. L'objectif de définir un compromis au cas par cas se dessine (voir notamment la sous-section 7.2.2.2).

7.1.2 Diagnosticabilité des systèmes

Le problème du diagnostic est par nature lié au problème de la diagnosticabilité. Contribuer sur le problème de diagnostic consiste à mettre en place des approches automatiques en s'appuyant sur une base de connaissances donnée et des observations disponibles. Même si l'approche de diagnostic met à profit toute la connaissance disponible, il n'y a pas de garantie que le résultat du diagnostic ne soit pas ambigu. L'analyse a priori de la diagnosticabilité répond à cette question : en fonction de la base de connaissance du système et des observations potentiellement disponibles peut-on garantir un résultat certain par une méthode de diagnostic appropriée ? C'est dans cette esprit que les travaux sur la diagnosticabilité ont été entrepris. En partant des systèmes statiques, la diagnosticabilité a été discutée et caractérisée en fonction des modes de fautes et de leurs extensions aux macromodes (section 2.3 page 32) ce qui a conduit également la définition des systèmes autoguérissants (section 5.1 page 153). Sur les systèmes à événements discrets où les travaux sur la diagnosticabilité sont les plus poussés dans la littérature, les travaux ont portés essentiellement sur deux aspects. Le premier d'entre eux est lié à la nature distribuée des systèmes, en développant des méthodes d'analyse de diagnosticabilité locale (section 3.3.3 page 88). Le deuxième a pour objectif d'augmenter l'expressivité des problèmes en ne s'attachant plus uniquement à des événements fautifs simples mais à des motifs d'intérêts (section 3.3.5 page 95). Ces travaux sur la diagnosticabilité ont enfin été mis à profit dans le cadre du diagnostic actif où l'objectif est de planifier un ensemble d'actions pour mener le système dans un état diagnosticable (section 5.2 page 161). Dans le cadre des systèmes temporisés, les contributions sur l'analyse de diagnosticabilité ont essentiellement porté sur l'analyse des capacités de diagnostic d'une méthode de reconnaissance de chroniques (section 4.3 page 136) avec la vérification de l'exclusivité d'un couple de chroniques qui est une condition nécessaire à la diagnosticabilité générale.

7.1.3 Diagnostic décentralisé

Mes travaux de thèse ont porté sur le diagnostic décentralisé de systèmes à événements discrets [32, 33, 43, 42, 46, 45, 44, 41]. Ces travaux ont été poursuivis en s'appuyant toujours sur l'idée qu'un traitement local du diagnostic sera toujours moins complexe qu'un traitement global. Dans cette optique, j'ai donc contribué dans un premier temps à une analyse de diagnosticabilité décentralisée dans les SEDs (section 3.3.3 page 88) mais également à étudier d'autres méthodes de diagnostic qui tendent à ne pas s'appuyer sur la connaissance globale du système (diagnostic par spécialisation de la section 3.4.2 page 108). Ces travaux ont également abouti à la définition d'une propriété propre à la décentralisation du diagnostic : la propriété de précision qui garantit des résultats locaux aussi précis que des résultats issus d'un raisonnement global (section 3.4.3 page 110).

7.1.4 Méta-diagnostic

Le méta-diagnostic est l'étude de la qualité d'un système de diagnostic. Au cours de ces travaux (section 2.4), nous avons pu mettre en évidence les problèmes auxquels on fait face lors d'un raisonnement automatique de diagnostic (qualité des connaissances, correction, complétude, qualité des observations, incertitude, perte, qualité des algorithmes de raisonnement, correction, complétude). À l'aide de cette théorie, nous disposons de moyens automatiques de déterminer si les raisonnements effectués sont corrects ou non et dans la négative, de localiser les éléments du raisonnements qui posent problème (connaissance, algorithmes, observations) en vue de les réparer.

7.1.5 Diagnostic et ses applications

Enfin, le raisonnement de diagnostic n'est pas une fin en soi, il sert à améliorer les décisions qu'elles soient humaines ou non. Une première étude a permis de caractériser des systèmes avec des capacités d'autoguérison (section 5.1 page 153). Les aspects diagnostic ont aussi été intégré dans une architecture de planification de mission dans laquelle on a pu définir des sessions de diagnostics actifs permettant de raffiner dans la mesure du possible les ambiguïtés sur l'état de santé de l'agent. Ce raffinement du diagnostic a pour conséquence une meilleure autonomie décisionnelle de l'agent (section 5.2 page 161). Le diagnostic sert également et bien évidemment à toute activité de maintenance. Dans cette optique, une formalisation générique des prérequis pour la définition correcte d'un problème de diagnostic décentralisé dans des systèmes hétérogènes pour l'aide à la maintenance classique a été proposée (section 5.3.2 page 173). Une extension pour la maintenance prévisionnelle a été également étudié intégrant le raisonnement diagnostic avec du pronostic de vieillissement (section 5.3.3 page 181). Un tel exemple d'intégration diagnostic-pronostic a été étudié par la suite (section 5.3.4 page 184).

7.2 Perspectives

7.2.1 Court terme

7.2.1.1 Étude du diagnostic de motifs temporels

Au cours de ce mémoire nous avons exposé les travaux sur le diagnostic et la diagnosticabilité de motifs atemporels dans les systèmes à événements discrets (section 3.3.5). Les travaux sur l'extension de ce problème aux motifs d'intérêts temporels dans les systèmes temporels sont en cours. *L'objectif est d'étendre l'analyse des motifs non seulement aux systèmes temporisés mais également d'étendre l'expressivité des motifs au temps.* La difficulté majeure quand on passe à la dimension temporelle et que

l'on peut faire face à des problèmes de décidabilité. Néanmoins il semble possible de caractériser des sous-classes de systèmes temporisés ainsi que des motifs de comportements temporisés pour lesquels la résolution du problème de diagnostic et de diagnosticabilité reste décidable. Ce travail en cours s'appuie bien évidemment sur l'outil de vérification de modèle TINA qui offre déjà un moyen de caractériser parmi l'ensemble des modèles et des systèmes à vérifier une sous-classe décidable et se rapproche du point de vue technique de l'analyse d'exclusivité des chroniques (section 4.2.4) qui s'appuie déjà sur le formalisme des réseaux de Petri temporels.

7.2.1.2 Synthèse de chroniques diagnosticables

Construire un diagnostiqueur qui prend en compte toutes les observations anticipées du système peut s'avérer combinatoirement complexe et donc en particulier sa mise en oeuvre peut ne pas être possible. Une façon d'abstraire le problème est de ne pas construire une machine qui suit pas à pas le système mais qui ne retient qu'un motif observable d'événements qui une fois reconnu porte une information de diagnostic. C'est notamment le cas des chroniques dans le cas temporisé mais on peut bien évidemment le considérer dans le cas atemporel où les chroniques deviennent dans ce cas des motifs d'événements observables muni de contraintes de précédence. L'idée consiste donc ici à construire à partir d'un modèle un ensemble de chroniques/motifs qui ont la propriété de surveiller des comportements diagnosticables. Ceci n'est possible que si on est en mesure d'effectuer une analyse de diagnosticabilité sur le modèle sous-jacent a priori. L'intérêt majeur de cette technique est de capturer la propriété de diagnosticabilité dans une base de chroniques ou de motifs et l'algorithme de diagnostic est dans ce cas réduit à la simple reconnaissance de chroniques qui garantissent la présence ou l'absence de motifs d'intérêts anticipés (diagnostic certifié, voir les perspectives moyen terme de la section 7.2.2.2).

7.2.1.3 Diagnostic actif distribué

Les travaux que nous avons développés sur le diagnostic actif partent du principe que le système est monolithique. Cette hypothèse est trop restrictive car elle induit là encore des problèmes de complexité spatiale et temporelle qui empêchent d'effectuer du diagnostic actif sur des systèmes complexes. *L'objectif ici est de distribuer cet algorithme de diagnostic actif en s'appuyant notamment sur l'analyse de diagnosticabilité locale sur chaque composant.* Dans un premier temps, on pourra considérer alors qu'un seul agent face à un système distribué propose un plan global d'actions. Puis, dans un second temps, l'objectif serait de mettre en place un ensemble d'agents planificateurs qui se coordonneraient afin de guider les composants dans un état où le diagnostic sera globalement raffiné. Pour atteindre cet objectif, on pourra s'appuyer sur les travaux précédents sur le diagnostic actif et sur l'analyse décentralisée de la diagnosti-

cabilité des systèmes avec également l'analyse de la précision des sous-systèmes. Un deuxième point essentiel concernant le diagnostic actif est que le plan (où la partie du plan) ne doit pas avoir trop d'influences sur l'objectif principal. L'objectif de diagnostic est toujours secondaire et ne doit servir que pour améliorer la décision globale (plan multi-objectifs).

7.2.1.4 Diagnostic à partir de modèles automates à contraintes

Les travaux présentés sur le diagnostic des systèmes à événements discrets s'appuient sur le fait que le comportement du système est totalement déterminé par l'occurrence d'événements si bien que sa représentation est faite sous la forme d'un système de transitions étiquetées par des événements. Que cela soit des automates ou des réseaux de Petri étiquetés, ces formalismes représentent des machines de type Mealy. Il existe cependant dans la littérature un autre jeu de travaux qui s'appuient uniquement sur les machines dites de Moore à savoir que l'information n'est pas sur la transition mais dans l'état ([Förstner et al., 2002, Lefebvre and Delherm, 2007]). Bien que ces deux représentations soient théoriquement équivalentes, elles expriment des points de vue différents sur le système sous-jacent qui sont exploités par des techniques de diagnostic indépendantes. Sur une machine de Mealy, l'observation est définie par l'occurrence d'événements alors que sur une machine de Moore l'observation est définie par la mesure d'une variable d'état. Également la notion de faute peut être représentée différemment et représenter différents phénomènes sur une machine de Moore (dérives) et sur une machine de Mealy (faute franche). *L'objectif de cet axe de recherche est de mêler dans une seule et même représentation la vision machine de Mealy et la vision machine de Moore.* Ce formalisme existe déjà et est le modèle sémantique des automates à contraintes [Brlek and Rauzy, 1994]. Ce formalisme est notamment utilisé par un langage formel qui se nomme Altarica [Point and Rauzy, 1999]. Le travail va donc consister à mettre en place une intégration des outils de diagnostic qui combinent les techniques de machine de Mealy et les techniques de machine de Moore pour être en mesure de résoudre tous les problèmes de diagnostic exprimable dans un langage de type Altarica. Un premier travail dans ce sens a déjà été défini dans [40]. On montre en effet que tous les problèmes de diagnostic définis dans le chapitre 1 mais également pour le chapitre 2 peuvent être représentés par un modèle Altarica. L'idée est donc de mettre en oeuvre une théorie générale qui intègre toutes les spécificités de ce formalisme et de mettre en oeuvre les outils en conséquence. Une extension temporelle pourrait également être envisagée dans un moyen terme.

7.2.1.5 Vers un déploiement optimal d'une architecture de diagnostic dans les SED

Par définition un système est constitué d'un ensemble de composants. Décentralisé par le fait qu'un système peut être localisé à différents endroits. S'ajoute à cela les problèmes de calcul en soi qui font qu'un algorithme de diagnostic global sur un système de n composants est exponentiel en n. Ce qui impose une décentralisation/distribution de l'algorithme comme je l'ai présenté par exemple dans mes travaux de thèse [41]. *Ainsi la question qui est posée est la suivante : étant donnés un système et sa topologie, étant données les capacités de calcul pour la surveillance de ce système, comment synthétiser l'architecture de diagnostic optimal pour ce système ?*. La question consiste à déterminer l'ensemble des diagnostiqueurs locaux qui aboutiront par communication à la construction d'un diagnostic précis pour tous les phénomènes que l'on cherche à anticiper dans le système tout en minimisant l'effort calculatoire. Ceci n'est possible qu'après une analyse de la précision des sous-systèmes en terme de diagnostic. Des éléments de réponse sont déjà proposés dans les travaux [29, 28].

7.2.2 Moyen terme

7.2.2.1 Diagnosticabilité : posons-nous la bonne question ?

La diagnosticabilité est la propriété d'un système qui garantit la certitude d'un résultat à partir de l'observabilité du système. Tous les travaux de la communauté se sont focalisés sur l'analyse effective de cette propriété pour un ensemble de fautes données, voire de motifs d'intérêt donnés. Très généralement, la réponse de l'analyse de diagnosticabilité est binaire à savoir : oui si le système est diagnosticable, non s'il ne l'est pas. La diagnosticabilité des systèmes est une propriété très forte qui impose beaucoup de restrictions, aussi les systèmes réels ne sont très vraisemblablement pas diagnosticables. Que dire et que faire si le résultat de l'analyse d'un système donné est négatif ? Avec un résultat négatif on sait seulement que le diagnostic pourra ne pas conclure avec certitude sur l'occurrence d'une faute ou d'un motif, autrement dit, un résultat négatif n'apporte aucune information pertinente sur la diagnosticabilité effective du système. L'analyse en cas de résultat négatif se doit donc d'être plus poussée comme je l'ai déjà suggéré dans les travaux ([36]) où le résultat de l'analyse négative était un langage observable ambigu. Il faut aller encore plus loin et proposer une analyse de diagnosticabilité qui soit plus contextuelle, plus informée afin de déterminer a priori les comportements diagnosticables et les comportements non diagnosticables du système. Ainsi, je me propose d'étendre les analyses de diagnosticabilité afin de rendre une information plus riche en cas de réponse négative et de mieux les exploiter au cours d'une session de diagnostic (voir les perspectives de la section 7.2.2.2).

Les travaux sur l'analyse de diagnosticabilité conduisent également à faire un deuxième constat. Avec l'extension de l'analyse de diagnosticabilité aux motifs de com-

portement, on a étendu l'analyse de diagnosticabilité sur un nombre infini de motifs possibles. On constate alors que ce qui importe n'est finalement pas de savoir si un événement ou un autre est diagnosticable mais il s'agit véritablement de définir l'ensemble des comportements (soit des événements simples, soit des motifs complexes) qui sont intrinsèquement diagnosticables dans le système. Une des perspectives sur l'analyse de diagnosticabilité va donc consister à redéfinir le problème en partant uniquement de l'observabilité du système pour en déterminer les comportements non observables qui sont diagnosticables. L'objectif est ainsi de définir le langage des motifs diagnosticables, qui dans un certain sens est le dual du langage contrôlable dans la théorie de la commande des systèmes type SED [Ramadge and Wonham, 1989]. Le travail sur le générateur aléatoire de systèmes diagnosticables [39] offre déjà une piste de travail intéressante pour répondre à la question. En effet, l'algorithme présenté dans ce générateur dispose d'éléments de connaissance sur les raisons pour lesquelles un comportement quelconque est ou n'est pas diagnosticable.

7.2.2.2 Adaptation en-ligne de l'algorithme de diagnostic à la diagnosticabilité courante

Dans la littérature, la plupart des travaux mettant en lien diagnostic et diagnosticabilité montrent seulement que si le système est diagnosticable alors il existe un diagnostiqueur qui par construction pourra déterminer avec certitude les fautes anticipées. Là encore, en relation avec la perspective précédente, le lien entre diagnostic et diagnosticabilité peut-être renforcé dès lors que l'analyse de diagnosticabilité en cas de réponse négative est plus informée. L'analyse de diagnosticabilité, qu'elle soit positive ou négative, apporte une information qui peut être exploitée en ligne lors de la phase de diagnostic. Étant donnée une situation où le diagnostic est ambigu, si je sais dans cette situation que les fautes anticipées ne sont pas diagnosticables, à quoi bon poursuivre le diagnostic ? De même si dans cette situation l'une des fautes est diagnosticable mais pas les autres (ou seules des macrofautes sont diagnosticables au sens des sections 2.3.2 ou encore 5.1.4 par exemple), pourquoi ne pas spécialiser le raisonnement pour le diagnostic de la faute diagnosticable (macrofautes) en question ? Derrière ces questions on retrouve l'intérêt de précompiler dans l'algorithme de diagnostic des informations de diagnosticabilité qui vont permettre d'augmenter la performance, l'efficacité du raisonnement en oubliant les objectifs de diagnostic pour lesquels nous sommes assurés qu'il n'y a pas de solution plus précise. Un autre point intéressant pour l'intégration de la connaissance de diagnosticabilité dans la construction de l'algorithme de diagnostic est que si la solution finale est effectivement ambiguë, la connaissance de diagnosticabilité embarquée dans l'algorithme de diagnostic va permettre de justifier ce résultat d'ambiguïté. Cette justification apporte une richesse supplémentaire à l'algorithme de diagnostic car elle permet d'expliquer à un opérateur ou un agent quelconque la raison pour laquelle on a une ambiguïté mais également l'information qui nous indique si

en attendant de nouvelles observations le résultat peut-être plus fin. Cette intégration entre diagnosticabilité et diagnostic a pour objectif d'aller vers une méthode de diagnostic certifié à savoir que le résultat obtenu peut être expliqué formellement par l'analyse de diagnosticabilité qui aura été effectuée au préalable sur un modèle formel. Afin d'aller dans cette direction, nous avons déjà des éléments. Un exemple d'intégration de l'algorithme de diagnostic et de l'analyse de diagnosticabilité se trouve dans les travaux sur le diagnostic actif (section 5.2). En effet dans ces travaux, on embarque déjà une analyse de discriminabilité sur le système en vue de mettre en place un plan d'actions qui raffine l'état de santé du système. D'autres travaux de la littérature vont également dans ce sens à savoir, en ligne au moment du diagnostic, la production d'informations de diagnosticabilité afin de savoir si la situation courante peut être raffinée ou non. La grande difficulté de cette intégration est la complexité calculatoire intrinsèque à la diagnosticabilité. Là encore, un compromis entre la complexité hors ligne et en ligne doit être trouvée. Dans cette même perspective, on peut également inclure la propriété de précision du diagnostic que l'on retrouve dans les systèmes distribués. En effet, en fonction des scénarios, les sous-systèmes, pouvant conduire à un diagnostic aussi précis que le diagnostic global, peuvent être différents. La précompilation d'une information sur la précision du diagnostic ne peut donc qu'améliorer les performances générales du raisonnement diagnostic sur le système.

7.2.2.3 Le bon algorithme à la bonne place

Par définition un système est constitué d'un ensemble de composants. Nous avons vu au cours de ce mémoire que ces composants peuvent être de nature hétérogène. La caractérisation logique formelle des prérequis pour la mise en place d'une architecture de diagnostic sur les systèmes hétérogènes dans la section 5.3.2. La nature même d'un système impose donc a minima une architecture décentralisée de diagnostic [41] ce que j'envisage d'étudier dans le court terme avec les systèmes type SED (section 7.2.1.5). Mais nous pouvons aller plus loin, supposons maintenant que le système n'est pas homogène. Certains composants peuvent avoir des dynamiques différentes. Il est possible que certains composants n'aient pas une dynamique qui soit véritablement en fonction du temps mais uniquement en fonction d'événements. La granularité temporelle entre deux composants n'est peut être pas la même du point de vue du diagnostic. Il peut y avoir des composants purement fonctionnels qui ont peu d'aspects temporels voire pas du tout. L'intégration de ces composants hétérogènes dans un système global, nécessite la même intégration de diagnostiqueurs locaux dans une architecture de diagnostic global, cette intégration étant guidée par des analyses de diagnosticabilité. Je me propose l'objectif de *déterminer une architecture de diagnostic où l'idée est de placer l'algorithme de diagnostic le plus adapté à un composant et de l'intégrer dans une architecture globale qui assure une cohérence globale du diagnostic du système.* L'idée ici est d'aller plus loin en généralisant les architectures de diagnostic pour qu'elles acceptent des composants

de différentes natures.

7.2.2.4 Adaptation des modèles de connaissance

L'ensemble des travaux qui ont été présentés dans ce mémoire reposent sur l'hypothèse de l'existence d'un modèle de comportement du système qui soit statique ou dynamique. La qualité du modèle a été discutée et analysée dans le cadre du système statique et de la théorie du méta diagnostic. Ce problème est évidemment extensible à tous les autres types de systèmes et en particulier aux systèmes à événements discrets et aux systèmes temporisés. Ce problème est lié à la manière, à la nature de l'acquisition des connaissances pour effectuer le raisonnement de diagnostic. De nos jours du fait de l'augmentation de la puissance de calcul des machines, les techniques d'apprentissage automatique sont de plus en plus présentes dans la littérature scientifique. Ce phénomène a évidemment des conséquences dans la communauté du diagnostic. Il existe en effet de nombreuses techniques pour apprendre des modèles à partir de flux d'observation en vue d'établir des modèles de diagnostic ou plus généralement des modèles de comportement du système. Ces méthodes type boîte noire ont l'intérêt de ne pas nécessiter d'un expert une connaissance a priori du système pour pouvoir établir des conclusions de diagnostic simplement en rapport avec les exécutions passées du système. La grande critique que l'on peut effectuer de ces méthodes repose sur la confiance que l'on peut porter à ces techniques. Chaque algorithme d'apprentissage par construction impose un biais et ce biais a des effets qui se répercutent sur le modèle appris à partir d'un flux d'observations donné. Ces méthodes ont donc intrinsèquement un problème d'explicabilité de leurs résultats.

Un de mes objectifs de recherche est effectivement de concilier les deux approches, nous appellerons cela une méthode *boîte grise* qui met en rapport un modèle dont la qualité est ce qu'elle est à un instant donné et une technique d'apprentissage pour raffiner ce modèle au cours du temps. *L'objectif est de définir des indicateurs de la qualité de la base de connaissances (voir en particulier ceux listés dans la théorie du méta-diagnostic) et de mettre en œuvre des techniques d'apprentissage automatique qui, par construction même de la méthode, garantissent une amélioration de la qualité de la base de connaissances (convergence, identification à la limite).* Dans cette vision, il existe ainsi toujours une distinction nette entre le raisonnement diagnostique qui s'appuie sur une base de connaissances, un modèle intelligible et donc explicable ; et les moyens d'acquérir ces types de modèle : soit par expertise, soit par apprentissage. Je pars du principe qu'il est illusoire de faire un véritable diagnostic sans aucune connaissance a priori du système. Ainsi dès lors que l'on cherche à faire un diagnostic on a nécessairement une connaissance a priori qui peut être parfois incomplète incorrecte mais néanmoins elle existe. Tout résultat de diagnostic repose sur cette connaissance et éventuellement un raffinement qui serait établi à l'aide d'un mécanisme d'apprentissage. Une piste de travail se trouve dans [14] où la contribution porte sur l'inférence supervisée et incrémentale

d'un modèle SED à partir de journaux et qui garantit un critère de convergence du modèle appris.

Chapitre 8

Bibliographie

8.1 Contributions

[1] Liliana Ardissono, Stefano Bocconi, Cinzia Cappiello, Luca Console, Marie-Odile Cordier, Johann Eder, Gerhard Friedrich, Mariagrazia Figini, Roberto Furnari, Anna Goy, Karim Guennoun, Vladimir Ivanchenko, Xavier Le Guillou, Stefano Modafferi, Enrico Mussi, Yannick Pencolé, Giovanna Petrone, Barbara Pernici, Claudia Picardi, Xavier Pucel, Filippo Ramoni, Marino Segnan, Audine Subias, Daniele Theiseder-Dupré, Louise Travé-Massuyès, and Thierry Vidal. Wsdiamond : an approach to web services - diagnosability, monitoring and diagnosis. In *eChallenges e-2007*, The Hague, Netherlands, 10 2007.

[2] Liliana Ardissono, Stefano Bocconi, Cinzia Cappiello, Luca Console, Marie-Odile Cordier, Johann Eder, Gerhard Friedrich, Mariagrazia Figini, Roberto Furnari, Anna Goy, Karim Guennoun, Vladimir Ivanchenko, Xavier Le Guillou, Stefano Modafferi, Enrico Mussi, Yannick Pencolé, Giovanna Petrone, Barbara Pernici, Claudia Picardi, Filippo Ramoni, Xavier Pucel, Marino Segnan, Audine Subias, Daniele Theiseder-Dupré, Louise Travé-Massuyès, and Thierry Vidal. Wsdiamond : an approach to web services - diagnosability, monitoring and diagnosis. *European Research Consortium for Informatics and Mathematics*, 70 :25–26, 7 2007.

[3] Liliana Ardissono, Stefano Bocconi, Cinzia Cappiello, Luca Console, Marie-Odile Cordier, Johann Eder, Gerhard Friedrich, Mariagrazia Figini, Anna Goy, Karim Guennoun, Vladimir Ivanchenko, Xavier Le Guillou, Stefano Modafferi, Enrico Mussi, Yannick Pencolé, Giovanna Petrone, Barbara Pernici, Claudia Picardi, Filippo Ramoni, Xavier Pucel, Marino Segnan, Audine Subias, Daniele Theiseder-Dupré, Louise Travé-Massuyès, and Thierry Vidal. *WS-DIAMOND : an approach to web services - DIAgnosability, MONitoring and Diagnosis*, pages 105–112. 2007.

[4] Nuno Belard, Michel Combacau, and Yannick Pencolé. Meta-diagnosis in fdi : Reasoning about false analytical redundancy relations. In *8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 379–384, Mexico, Mexico, 8 2012.

[5] Nuno Belard, Yannick Pencolé, and Michel Combacau. Defining and exploring properties in diagnostic systems. In *21st International Workshop on Principles of Diagnosis*, pages 161–168, Portland (OR), United States, 10 2010.

[6] Nuno Belard, Yannick Pencolé, and Michel Combacau. Medito : a logic-based meta diagnosis tool. In *IEEE International Conference on Tools with Artificial Intelligence*, pages 709–716, Boca Raton (FL), United States, 11 2011.

[7] Nuno Belard, Yannick Pencolé, and Michel Combacau. Théorie de méta-diagnostic : raisonnement sur les systèmes de diagnostic. In *Journées de l'Intelligence Artificielle Fondamentale*, Lyon, France, 6 2011.

[8] Nuno Belard, Yannick Pencolé, and Michel Combacau. A theory of meta-diagnosis : reasoning about diagnostic systems. In *22nd International Joint Conference on Artificial Intelligence*, pages 731–737, Barcelona, Spain, 2011.

[9] Elodie Chanthery and Yannick Pencolé. Modélisation et intégration du diagnostic actif dans une architecture embarquée. *Journal européen des systèmes automatisés*, 43 :789–803, 2009.

[10] Elodie Chanthery and Yannick Pencolé. Monitoring and active diagnosis for discrete-event systems. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Barcelona, Spain, 7 2009.

[11] Elodie Chanthery and Yannick Pencolé. Principles of self-maintenance in an on-board architecture including active diagnosis. In *The IJCAI-09 Workshop on Self-* and Autonomous Systems : reasoning and integration challenges*, pages 43–50, Pasadena, California, United States, 7 2009.

[12] Elodie Chanthery, Yannick Pencolé, and Nicolas Bussac. An ao*-like algorithm implementation for active diagnosis. In *10th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, pages 378–385, Sapporo, Japan, 8 2010.

[13] Elodie Chanthery, Yannick Pencolé, Pauline Ribot, and Louise Travé-Massuyès. Hydiag : extended diagnosis and prognosis for hybrid systems. In *26th international workshop on Principles of Diagnosis*, pages 281–284, Paris, France, 8 2015.

[14] Cody Christopher, Yannick Pencolé, and Alban Grastien. Inference of fault signatures of discrete-event systems from event logs. In *28th International Workshop on Principles of Diagnosis*, pages 219–233, Brescia, Italy, 1 2018.

[15] Michel Combacau and Yannick Pencolé. Agreed definitions for archistic project. wp1. fundamentals and requirements. Technical Report 7188, Laboratoire d'Analyse et d'Architecture de Systèmes, 2007.

[16] Michel Combacau, Yannick Pencolé, and Pauline Ribot. Deliverable wp4 projet archistic - logical characterisation of the diagnostic problem. Technical report, Laboratoire d'Analyse et d'Architecture de Systèmes, 2008.

[17] Luca Console, Danilo Ardagna, Liliana Ardissono, Stefano Bocconi, Cinzia Cappiello, Marie-Odile Cordier, Philippe Dague, Khalil Drira, Johann Eder, Gerhard Friedrich, Mariagrazia Figini, Roberto Furnari, Anna Goy, Karim Guennoun, Andreas Hess, Vladimir Ivanchenko, Xavier Le Guillou, Marek Lehmann, Yingmin Li, Jürgen Mangler, Tarek Melliti, Stefano Modafferi, Enrico Mussi, Yannick Pencolé, Giovanna Petrone, Barbara Pernici, Claudia Picardi, Xavier Pucel, Sophie Robin, Laurence Rozé, Marino Segnan, Nick Amirreza Tahamtan, Annette ten Teije, Daniele Theiseder-Dupré, Louise Travé-Massuyès, Frank van Harmelen, and Thierry Vidal. Ws-diamond : Web services - diagnosability, monitoring and diagnosis. In *18th International Workshop on Principles of Diagnosis*, pages 243–250, Nashville, Tennessee, United States, 5 2007.

[18] Luca Console, Danilo Ardagna, Liliana Ardissono, Stefano Bocconi, Cinzia Cappiello, Marie-Odile Cordier, Khalil Drira, Johann Eder, Gerhard Friedrich, Mariagrazia Figini, Roberto Furnari, Anna Goy, Karim Guennoun, Andreas Hess, Vladimir Ivanchenko, Xavier Le Guillou, Marek Lehmann, Jürgen Mangler, Yingmin Li, Tarek Melliti, Stefano Modafferi, Enrico Mussi, Yannick Pencolé, Giovanna Petrone, Barbara Pernici, Claudia Picardi, Xavier Pucel, Sophie Robin, Laurence Rozé, Marino Segnan, Nick Amirreza Tahamtan, Annette ten Teije, Daniele Theiseder-Dupré, Louise Travé-Massuyès, Frank van Harmelen, Thierry Vidal, and Audine Subias. *WS-DIAMOND web services - DIAgnosability, MONitoring and Diagnosis*, pages 213–240. 2009.

[19] Marie-Odile Cordier, Yannick Pencolé, Louise Travé-Massuyès, and Thierry Vidal. Caractérisation des systèmes autoguérissants : diagnostiquer ce que l'on peut réparer. *Information-Interaction-Intelligence*, 8(2) :123–151, 2008.

[20] Mickael Dievart, Xavier Desforges, Philippe Charbonnaud, Bernard Archimède, Michel Combacau, Yannick Pencolé, and Pauline Ribot. Archistic project. wp3. preliminary process and software architecture. Technical Report 8394, Laboratoire d'Analyse et d'Architecture de Systèmes, 2008.

[21] Khalil Drira, Karim Guennoun, Francisco José Moo-Mena, Yannick Pencolé, Xavier Pucel, Audine Subias, and Louise Travé-Massuyès. Requirements, application scenarios, overall architecture, and test/validation specification, common working environment and standards at milestone m1. Technical Report 6879, Laboratoire d'Analyse et d'Architecture de Systèmes, 2006.

[22] Houssam-Eddine Gougam, Yannick Pencolé, and Audine Subias. Diagnosability analysis of patterns on bounded labeled prioritized petri nets. *Journal of Discrete Event Dynamic Systems : Theory and Applications*, 27(1) :143–180, 3 2017.

[23] Houssam-Eddine Gougam, Audine Subias, and Yannick Pencolé. Timed diagnosability analysis based on chronicles. In *8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 1256–1261, Mexico, Mexico, 8 2012.

[24] Houssam-Eddine Gougam, Audine Subias, and Yannick Pencolé. Diagnosticabilité de motifs de supervision par dépliage de réseaux de petri. In *Journées Doctorales Journées Nationales MACS*, Strasbourg, France, 7 2013.

[25] Houssam-Eddine Gougam, Audine Subias, and Yannick Pencolé. Supervision patterns : Formal diagnosability checking by petri net unfolding. In *4th IFAC Workshop on Dependable Control of Discrete Systems*, pages 73–78, York, United Kingdom, 9 2013.

[26] Houssam-Eddine Gougam, Audine Subias, and Yannick Pencolé. Discriminability analysis of supervision patterns by net unfoldings. In *12th IFAC - IEEE International Workshop on Discrete Event Systems*, pages 459–464, Cachan, France, 5 2014.

[27] Carine Jauberthie, Yannick Pencolé, Renaud Pons, Pauline Ribot, and Louise Travé-Massuyès. Diagnosis and prognosis in health monitoring systems. state of the art. Technical Report 12680, Laboratoire d'Analyse et d'Architecture de Systèmes, 2012.

[28] Priscilla Kan John, Alban Grastien, and Yannick Pencolé. Synthesis of a distributed and accurate diagnoser. In *21st International Workshop on Principles of Diagnosis*, pages 209–216, Portland (OR), United States, 10 2010.

[29] Priscilla Kan John, Alban Grastien, Yannick Pencolé, and Pauline Ribot. Synthèse d'un diagnostiqueur distribué et précis. In *17ème congrès francophone AFRIF-AFIA Reconnaissance des Formes et Intelligence Artificielle*, pages 646–653, Caen, France, 1 2010.

[30] Euriell Le Corronc, Alexandre Sahuguède, and Yannick Pencolé. Détection et localisation de fautes temporelles dans les systèmes (max,+)-linéaires. In *11ème Colloque sur la Modélisation des Systèmes Réactifs*, Marseille, France, 11 2017.

[31] Ghyslain Maitre, Yannick Pencolé, Audine Subias, and Houssam-Eddine Gougam. Modélisation et analyse de chroniques pour le diagnostic. In *Modélisation des Systèmes Réactifs*, Nancy, France, 11 2015.

[32] Yannick Pencolé. Approche diagnostiqueur décentralisé : application aux réseaux de télécommunications. In *5èmes Rencontres nationales des Jeunes Chercheurs en Intelligence Artificielle*, pages 309–322, Lyon, France, 9 2000.

[33] Yannick Pencolé. Decentralized diagnoser approach : application to telecommunication networks. In *11th International Workshop on Principles of Diagnosis*, pages 185–192, Morelia, Mexico, 2000.

[34] Yannick Pencolé. Diagnosability analysis of distributed discrete event systems. In *16th European Conference on Artificial Intelligence*, pages 43–47, Valencia, Spain, 8 2004.

[35] Yannick Pencolé. Diagnosability analysis of distributed discrete event systems. In *15th International Workshop on Principles of Diagnosis*, pages 173–178, Carcassonne, France, 6 2004.

[36] Yannick Pencolé. Assistance for the design of a diagnosable component-based system. In *17th IEEE International Conference on Tools with Artificial Intelligence*, pages 549–556, Hong-Kong, Hong Kong, 11 2005.

[37] Yannick Pencolé. Fault diagnosis in discrete-event systems : How to analyse algorithm performance ? In *Diagnostic reasoning : Model Analysis and Performance*, pages 19–25, Montpellier, France, 8 2012.

[38] Yannick Pencolé. Dito : a csp-based diagnostic engine. In *21st European Conference on Artificial Intelligence*, pages 699–704, Prague, Czech Republic, 8 2014.

[39] Yannick Pencolé. Random generator of k diagnosable discrete event systems. In *26th international workshop on Principles of Diagnosis*, pages 277–280, Paris, France, 8 2015.

[40] Yannick Pencolé, Elodie Chanthery, and Thierry Peynot. Definition of model-based diagnosis problems with altarica. In *27th International Workshop on Principles of Diagnosis*, Denver, Colorado, United States, 10 2016.

[41] Yannick Pencolé and Marie-Odile Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164(2) :121–170, 5 2005.

[42] Yannick Pencolé, Marie-Odile Cordier, and Laurence Rozé. A decentralized model-based diagnostic tool for complex systems. In *13th International Conference on Tools with Artificial Intelligence*, pages 95–102, Dallas, TX, United States, 11 2001.

[43] Yannick Pencolé, Marie-Odile Cordier, and Laurence Rozé. Incremental decentralized diagnosis approach for the supervision of a telecommunication network. In *12th International Workshop on Principles of Diagnosis*, pages 151–158, Via Lattea, Italy, 3 2001.

[44] Yannick Pencolé, Marie-Odile Cordier, and Laurence Rozé. A decentralized model-based diagnostic tool for complex systems. *International Journal on Artificial Intelligence Tools*, 11(3) :327–346, 9 2002.

[45] Yannick Pencolé, Marie-Odile Cordier, and Laurence Rozé. Incremental decentralized diagnosis approach for the supervision of a telecommunication network. In *41st IEEE Conference on Decision and Control*, pages 435–440, Las Vegas, NV, United States, 12 2002.

[46] Yannick Pencolé, Marie-Odile Cordier, and Laurence Rozé. Une stratégie efficace pour une approche décentralisée du diagnostic de systèmes complexes. In *13ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle*, pages 259–267, Angers, France, 1 2002.

[47] Yannick Pencolé, Romain Pichard, and Pierre Fernbach. Modular fault diagnosis in discrete-event systems with a cpn diagnoser. In *9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, pages 470–475, Paris, France, 9 2015.

[48] Yannick Pencolé, Anika Schumann, and Dmitry Kamenetsky. Towards low-cost fault diagnosis in large component-based systems. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 1473–1478, Beijing, China, 8 2006.

[49] Yannick Pencolé and Audine Subias. A chronicle-based diagnosability approach for discrete timed-event systems : application to web-services. *Journal of Universal Computer Science*, 15(17) :3246–3272, 11 2009.

[50] Xavier Pucel, Louise Travé-Massuyès, and Yannick Pencolé. Another point of view on diagnosability. In *19th International Workshop on Principles of Diagnosis*, pages 331–338, Blue Mountains, New South Wales, Australia, 9 2008.

[51] Xavier Pucel, Louise Travé-Massuyès, and Yannick Pencolé. Another point of view on diagnosability. In *4th Starting Artificial Intelligence Researchers' Symposium*, pages 151–162, Patras, Greece, 7 2008.

[52] Pauline Ribot, Yannick Pencolé, and Michel Combacau. Archistic project, wp2 deliverable. diagnosis and prognosis - state of the art. Technical Report 7184, Laboratoire d'Analyse et d'Architecture de Systèmes, 2007.

[53] Pauline Ribot, Yannick Pencolé, and Michel Combacau. Characterization of requirements and costs for the diagnosability of distributed discrete event systems. In *5th Workshop on Advanced Control and Diagnosis*, Grenoble, France, 11 2007.

[54] Pauline Ribot, Yannick Pencolé, and Michel Combacau. Deliverable wp5 projet archistic - solutions for failure prognosis - prognostic function specification. Technical report, Laboratoire d'Analyse et d'Architecture de Systèmes, 2008.

[55] Pauline Ribot, Yannick Pencolé, and Michel Combacau. Deliverable wp9.1 projet archistic - fault diagnosability assurance - diagnosis performance criteria. Technical report, Laboratoire d'Analyse et d'Architecture de Systèmes, 2008.

[56] Pauline Ribot, Yannick Pencolé, and Michel Combacau. Design requirements for the diagnosability of distributed discrete event systems. In *19th International Workshop on Principles of Diagnosis*, pages 347–354, Blue Mountains, New South Wales, Australia, 9 2008.

[57] Pauline Ribot, Yannick Pencolé, and Michel Combacau. Diagnosis and prognosis for the maintenance of complex systems. In *International Conference on Prognostics and Health Management 2008*, Denver, Colorado, United States, 10 2008.

[58] Pauline Ribot, Yannick Pencolé, and Michel Combacau. Diagnosis and prognosis for the maintenance of complex systems. In *IEEE International Conference on Systems, Man and Cybernetics, 2009.*, pages 4146–4151, San Antonio, Texas, United States, 10 2009.

[59] Pauline Ribot, Yannick Pencolé, and Michel Combacau. Functional prognostic architecture for the maintenance of complex systems. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Barcelona, Spain, 6 2009.

[60] Pauline Ribot, Yannick Pencolé, and Michel Combacau. Generic characterization of diagnosis and prognosis for complex heterogeneous systems. *International Journal of Prognostics and Health Management*, 4(2), 2013.

[61] Alexandre Sahuguède, Euriell Le Corronc, and Yannick Pencolé. Design of indicators for the detection of time shift failures in (max, +)-linear systems. In *20th World Congress of the International Federation of Automatic Control*, Toulouse, France, 7 2017.

[62] Anika Schumann and Yannick Pencolé. Scalable diagnosability checking of event-driven systems. In *20th International Joint Conference on Artificial Intelligence*, pages 575–580, Hyderabad, India, 1 2007.

[63] Anika Schumann, Yannick Pencolé, and Sylvie Thiébaux. Diagnosis of discrete-event systems using bdds. In *15th International Workshop on Principles of Diagnosis*, pages 197–202, Carcassonne, France, 6 2004.

[64] Anika Schumann, Yannick Pencolé, and Sylvie Thiébaux. Symbolic models for diagnosing discrete-event systems. In *16th European Conference on Artificial Intelligence*, pages 1085–1086, Valencia, Spain, 8 2004.

[65] Anika Schumann, Yannick Pencolé, and Sylvie Thiébaux. A spectrum of symbolic on-line diagnosis approaches. In *Twenty-Second Conference on Artificial Intelligence*, pages 335–340, Vancouver, Canada, 2007.

[66] Anika Schumann, Yannick Pencolé, and Sylvie Thiébaux. A decentralised symbolic diagnosis approach. In *19th European Conference on Artificial Intelligence*, pages 99–104, Lisbon, Portugal, 8 2010.

[67] Xingyu Su, Alban Grastien, and Yannick Pencolé. Window-based diagnostic algorithms for discrete event systems : What information to remember. In *25th International Workshop on Principles of Diagnosis*, Graz, Austria, 11 2014.

[68] Louise Travé-Massuyès, Renaud Pons, Pauline Ribot, Yannick Pencolé, and Carine Jauberthie. Condition-based monitoring and prognosis in an error-bounded

framework. In *26th international workshop on Principles of Diagnosis*, pages 83–90, Paris, France, 8 2015.

[69] Yuhong Yan, Marie-Odile Cordier, Yannick Pencolé, and Alban Grastien. Monitoring web service networks in a model-based approach. In *Third European Conference on Web Services*, pages 192–203, Växjö, Sweden, 11 2005.

[70] Yuhong Yan, Philippe Dague, Yannick Pencolé, and Marie-Odile Cordier. A model-based approach for diagnosing fault in web service processes. *International Journal of Web Services Research*, 6(1) :87–110, 2009.

8.2 Références

[afn,] Norme afnor nf en 13306, terminologie de la maintenance. Technical Report 13306.

[Adrot et al., 1999] Adrot, O., Ragot, J., and Maquin, D. (1999). Fault detection with model parameter structured uncertainties. In *European Control Coference*, pages 475–480, Karlsruhe, Germany.

[Alami et al., 1998] Alami, R., Chatila, R., Fleury, S., Ghallab, M., and Ingrand, F. (1998). An architecture for autonomy. *International Journal of Robotics Research*, 17 :315–337.

[Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11) :832–843.

[Allen, 1984] Allen, J. F. (1984). Towards a general theory of action and time. *Artificial Intelligence*, 23(2) :123–154.

[Arnold, 1992] Arnold, A. (1992). *Systèmes de transitions finis et sémantique des processus communicants*. Dunod.

[Baccelli et al., 1992] Baccelli, F., Cohen, G., Olsder, G. J., and Quadrat, J.-P. (1992). *Synchronization and Linearity An Algebra for Discrete Event Systems*. John Wiley & Sons, Inc.

[Baier and Katoen, 2008] Baier, C. and Katoen, J.-P. (2008). *Principles of Model Checking*. MIT Press.

[Barbier et al., 2006] Barbier, M., Gabard, J.-F., Vizcaino, D., and Bonnet-Torrès, O. (2006). Procosa : a software package for autonomous system supervision. In *First National Workshop on Control Architectures of Robots*, Montpellier, France.

[Baroni et al., 1999] Baroni, P., Lamperti, G., Pogliano, P., and Zanella, M. (1999). Diagnosis of large active systems. *Artificial Intelligence*, 110(1) :135–183.

[Basile et al., 2014] Basile, F., Cabasino, M. P., and Seatzu, C. (2014). State estimation and fault diagnosis of labeled time petri net systems with unobservable transitions. *Transactions on Automatic Control*, 60(4) :997–1009.

[Basile et al., 2009] Basile, F., Chiacchio, P., and De Tommasi, G. (2009). An efficient approach for online diagnosis of discrete event systems. *Transactions on Automatic Control*, 54(4) :748–759.

[Basile et al., 2012] Basile, F., Chiacchio, P., and De Tommasi, G. (2012). On k-diagnosability of petri nets via integer linear programming. *Automatica*, 48(9) :2047–2058.

[Bayoudh et al., 2008] Bayoudh, M., Travé-Massuyès, L., and Olive, X. (2008). Towards active diagnosis of hybrid systems. In *19th International Workshop on Principles of Diagnosis*, pages 231–238, Blue Mountains, New South Wales, Australia.

[Beldiceanu et al., 2007] Beldiceanu, N., Carlsson, M., Demassey, S., and Petit, T. (2007). Global constraint catalogue : Past, present and future. *Constraints*, 12(1) :21–62.

[Benveniste et al., 2003] Benveniste, A., Fabre, E., Haar, S., and Jard, C. (2003). Diagnosis of asynchronous discrete-event systems : a net unfolding approach. *Transactions on Automatic Control*, 48(5) :714–727.

[Bérard et al., 2005] Bérard, B., Cassez, F., Haddad, S., Lime, D., and Roux, O. H. (2005). *Comparison of different semantics for time Petri nets*, pages 293–307.

[Berthomieu et al., 2007] Berthomieu, B., Peres, F., and Vernadat, F. (2007). Model-checking bounded prioriterized time petri nets. In *15th International Symposium on Automated Technology for Verification and Analysis, LNCS4762*, pages 523–532, Tokyo, Japan.

[Berthomieu et al., 2004] Berthomieu, B., Ribet, P. O., and Vernadat, F. (2004). The tool tina ? construction of abstract state spaces for petri nets and time petri nets. *International Journal of Production Research*, 42(14) :2741–2756.

[Biswas and Manders, 2006] Biswas, G. and Manders, E. (2006). Integrated systems health management to achieve in complex systems. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 1207–1212, Beijing, China.

[Boel and Jiroveanu, 2013] Boel, R. K. and Jiroveanu, G. (2013). *The On-Line Diagnosis of Time Petri Nets*, pages 343–364.

[Bonet and Geffner, 2000] Bonet, B. and Geffner, H. (2000). Planning with incomplete information as heuristic search in belief space. In *Fifth International Conference on Artificial Intelligence Planning Systems*, pages 52–61, Breckenridge, United States.

[Bonhomme, 2015] Bonhomme, P. (2015). Fault diagnosis of p-time labeled petri net systems. In *9th Workshop on Verification and Evaluation of Computer and Communication Systems*, pages 11–22, Bucharest, Romania.

[Boufaied et al., 2004] Boufaied, A., Subias, A., and Combacau, M. (2004). Distributed fault detection with delays consideration. In *8th World Multi-Conference on Systemics, Cybernetics and Informatics*, pages 135–140, Orlando, Florida, United States.

[Bouyer et al., 2005] Bouyer, P., Chevalier, F., and D'Souza, D. (2005). Fault diagnosis using timed automata. In *8th international conference on Foundations of Software Science and Computation Structures*, pages 219–233, Edinburgh, United Kingdom.

[Branke et al., 2004] Branke, J., Deb, K., and Steuer, R. E. (2004). Practical approaches to multi-objective optimization. In *Dagstuhl Seminar*, Dagstuhl, Germany.

[Brlek and Rauzy, 1994] Brlek, S. and Rauzy, A. (1994). Synchronization of constrained transitions systems. In *1st International Symposium on Parallel Symbolic Computation*, pages 54–62, HagenbergLinz, Austria.

[Brotherton et al., 2000] Brotherton, T., Jahns, G., Jacobs, J., and Wroblewski, D. (2000). Prognosis of faults in gas turbine engines. In *IEEE Aerospace conference*, pages 163–171, Big Sky, Montana, United States.

[Bryant, 1986] Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8) :677–691.

[Byington et al., 2004] Byington, C., Kalgren, P., Dunkin, B., and Donovan, B. (2004). Advanced diagnostic/prognostic reasoning and evidence transformation techniques for improved avionics maintenance. In *IEEE Aerospace Conference*, pages 3424–3434, Big Sky, Montana, United States.

[Cabasino et al., 2012] Cabasino, M. P., Giua, A., Lafortune, S., and Seatzu, C. (2012). A new approach for diagnosability analysis of petri nets using verifier nets. *Transactions on Automatic Control*, 57(12) :3104–3117.

[Cabasino et al., 2009] Cabasino, M. P., Giua, A., and Seatzu, C. (2009). Diagnosability of bounded petri nets. In *48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, pages 1254–1260, Shanghai, China.

[Cabasino et al., 2010] Cabasino, M. P., Giua, A., and Seatzu, C. (2010). Fault detection for discrete event systems using petri nets with unobservable transitions. *Automatica*, 46(9) :1531–1539.

[Cabasino et al., 2014] Cabasino, M. P., Giua, A., and Seatzu, C. (2014). Diagnosability of discrete-event systems using labeled petri nets. *IEEE Transactions on Automation Science and Engineering*, 11(1) :144–153.

[Camci et al., 2007] Camci, F., Valentine, G., and Navarra, K. (2007). Methodologies for integration of phm systems with maintenance data. In *IEEE Aerospace Conference*, pages 4110–4118, Big Sky, Montana, United States.

[Carrault et al., 1999] Carrault, G., Cordier, M.-O., Quiniou, R., Garreau, M., Bellanger, J.-J., and Bardou, A. (1999). A model-based approach for learning to identify cardiac arrhythmias. In *Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making*, pages 165–174, Aalborg, Denmark.

[Cassandras and Lafortune, 2008] Cassandras, C. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer.

[Cassez et al., 2007] Cassez, F., Tripakis, S., and Altisen, K. (2007). Sensor minimization problems with static or dynamic observers for fault diagnosis. In *7th International Conference on Application of Concurrency to System Design*, pages 90–99, Bratislava, Slovakia.

[Chaki et al., 2004] Chaki, S., Clarke, E. M., Ouaknine, J., Sharygina, N., and Sinha, N. (2004). State/event-based software model checking. In *14th International Conference on Integrated Formal Methods*, pages 128–147, Canterbury, United Kingdom.

[Chanthery et al., 2005] Chanthery, E., Barbier, M., and Farges, J.-L. (2005). *Integration of Mission Planning and Flight Scheduling for Unmanned Aerial Vehicles*, pages 109–118.

[Chatain and Jard, 2005] Chatain, T. and Jard, C. (2005). Time supervision of concurrent systems using symbolic unfoldings of time petri nets. In *3rd International Conference on Formal Modeling and Analysis of Timed Systems*, pages 196–210, Uppsala, Sweden.

[Chen and Patton, 1999] Chen, J. and Patton, R. (1999). *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Springer.

[Cheriere, 2009] Cheriere, V. (2009). Monitoring, diagnosis and architecture guidelines for the design of health monitoring agents. Technical Report 831919, Airbus.

[Chittaro et al., 1993] Chittaro, L., Guida, G., Tasso, C., and Toppano, E. (1993). Functional and teleological knowledge in the multimodeling approach for reasoning about physical systems : a case study in diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6) :1718–1751.

[Choppy et al., 2009] Choppy, C., Bertrand, O., and Carle, P. (2009). Coloured petri nets for chronicle recognition. In *International Conference on Reliable Software Technologies*, pages 266–281, Brest, France.

[Cimatti et al., 2003] Cimatti, A., Pecheur, C., and Cavada, R. (2003). Formal verification of diagnosability via symbolic model checking. In *18th International Joint Conference on Artificial Intelligence*, pages 363–369, Acapulco, Mexico.

[Clarke et al., 1999] Clarke, E. M., Grumberg, O., and Peled, D. (1999). *Model Checking*. MIT Press.

[Console et al., 2000] Console, L., Picardi, C., and Ribaudo, M. (2000). Diagnosis and diagnosability analysis using pepa. In *14th European Conference on Artificial Intelligence*, pages 131–135, Berlin, Germany.

[Console et al., 1989] Console, L., Theiseder-Dupré, D., and Torasso, P. (1989). A theory of diagnosis for incomplete causal models. In *11th International Joint Conference on Artificial Intelligence*, pages 1311–1317, Detroit, Michigan, United States.

[Console and Torasso, 1990] Console, L. and Torasso, P. (1990). Integrating models of the correct behavior into abductive diagnosis. In *9th European Conference on Artificial Intelligence*, pages 160–166, Stockholm, Sweden.

[Console and Torasso, 1991] Console, L. and Torasso, P. (1991). A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7(3) :133–141.

[Cordier et al., 2000a] Cordier, M.-O., Dague, P., Dumas, M., Lévy, F., Montmain, J., Staroswiecki, M., and Travé-Massuyès, L. (2000a). Ai and automatic control theory approaches of model-based diagnosis : links and underlying hypotheses. In *Fault Detection, Supervision and Safety for Technical Processes*, pages 279–284, Budapest, Hungary.

[Cordier et al., 2000b] Cordier, M.-O., Dague, P., Dumas, M., Lévy, F., Montmain, J., Staroswiecki, M., and Travé-Massuyès, L. (2000b). A comparative analysis of ai and control theory approaches to model-based diagnosis. In *14th European Conference on Artificial Intelligence*, pages 136–140, Berlin, Germany.

[Cordier and Dousson, 2000] Cordier, M.-O. and Dousson, C. (2000). Alarm driven monitoring based on chronicles. In *Fault Detection, Supervision and Safety for Technical Processes*, pages 286–291, Budapest, Hungary.

[Cordier et al., 2006] Cordier, M.-O., Travé-Massuyès, L., and Pucel, X. (2006). Comparing diagnosability in continous and discrete-event systems. In *17th International Workshop on Principles of Diagnosis*, pages 55–60, Burgos, Spain.

[Coste-Marquis and Marquis, 1998] Coste-Marquis, S. and Marquis, P. (1998). Characterizing consistency-based diagnoses. In *5th International Symposium on Artificial Intelligence and Mathematics*.

[Cottenceau et al., 2000] Cottenceau, B., Lhommeau, M., Hardouin, L., and Boimond, J.-L. (2000). Data processing tool for calculation in dioid. In *5th Workshop on Discrete Event Systems*, Ghent, Belgium.

[Cram et al., 2011] Cram, D., Mathern, B., and Mille, A. (2011). A complete chronicle discovery approach : application to activity analysis. *Expert Systems*, 4(29) :321–346.

[Daigle and Goebel, 2011] Daigle, M. J. and Goebel, K. (2011). A model-based prognostics approach applied to pneumatic valves. *International Journal of Prognostics and Health Management*, 2(8).

[Darwiche and Marquis, 2002] Darwiche, A. and Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17 :229–264.

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

[Debouk et al., 1999] Debouk, R., Lafortune, S., and Teneketzis, D. (1999). On an optimization problem in sensor selection for failure diagnosis. In *38th Conference on Decision and Control*, pages 4990–4995, Phoenix, Arizona, United States.

[Debouk et al., 2002a] Debouk, R., Lafortune, S., and Teneketzis, D. (2002a). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Journal of Discrete Event Dynamic Systems : Theory and Applications*, 10 :33–86.

[Debouk et al., 2002b] Debouk, R., Lafortune, S., and Teneketzis, D. (2002b). On an optimization problem in sensor selection. *Journal of Discrete Event Dynamic Systems : Theory and Applications*, 12(4) :417–445.

[Dechter et al., 1991] Dechter, R., Meiry, I., and Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, 49 :61–95.

[deKleer, 1976] deKleer, J. (1976). Local methods for localizing faults in electronic circuits. Technical Report 394, Artificial Intelligence Laboratory (MIT).

[deKleer et al., 1992] deKleer, J., Mackworth, A., and Reiter, R. (1992). Characterizing diagnoses and systems. *Artificial Intelligence*, 56 :197–222.

[deKleer and Williams, 1987] deKleer, J. and Williams, B. C. (1987). Diagnosing multiple faults. *Artificial Intelligence*, 32 :97–130.

[deKleer and Williams, 1989] deKleer, J. and Williams, B. C. (1989). Diagnosis with behavioral modes. In *11th international joint conference on Artificial intelligence*, pages 1324–1330, Detroit, MI, United States.

[Dousson, 2002] Dousson, C. (2002). Extending and unifying chronicle representation with event counters. In *15th European Conference on Artificial Intelligence*, pages 257–261, Lyon, France.

[Dousson and Duong, 1999a] Dousson, C. and Duong, T. V. (1999a). Découverte de chroniques avec contraintes temporelles à partir de journaux d ?alarmes pour la supervision de réseaux de télécommunications. In *Première Conférence francophone sur l ?Apprentissage automatique*, pages 17–24, Palaiseau, France.

[Dousson and Duong, 1999b] Dousson, C. and Duong, T. V. (1999b). Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In *16th International Joint Conference on Artificial Intelligence*, pages 620–626, Stockholm, Sweden.

[Dousson et al., 1993] Dousson, C., Gaborit, P., and Ghallab, M. (1993). Situation recognition : representation and algorithms. In *13th International Joint Conference on Artificial Intelligence*, pages 166–172, Chambéry, France.

[Dressler and Struss, 1994] Dressler, O. and Struss, P. (1994). Model-based diagnosis with the default-based diagnosis engine : Effective control strategies that work in practice. In *11th European Conference on Artificial Intelligence*, pages 677–681, Amsterdam, Netherlands.

[Engel et al., 2000] Engel, S., Gilmartin, B., Bongsort, K., and Hess, A. (2000). Prognostics, the real issues involved with predicting life remaining. In *IEEE Aerospace conference*, pages 457–469, Big Sky, Montana, United States.

[Esparza et al., 2002] Esparza, J., Römer, S., and Vogler, W. (2002). An improvement of mcmillan's unfolding algorithm. *Formal Methods in System Design*, 20(3) :285–310.

[Fabre et al., 2005] Fabre, E., Benveniste, A., Haar, S., and Jard, C. (2005). Distributed monitoring of concurrent and asynchronous systems. *Journal of Discrete Event Dynamic Systems : Theory and Applications*, 15 :33–83.

[Fabre et al., 2004] Fabre, E., Benveniste, A., Haar, S., Jard, C., and Aghasaryan, A. (2004). Algorithms for distributed fault management in telecommunications networks. In *11th International Conference on Telecommunications*, pages 820–825, Fortaleza, Brazil.

[Feldman et al., 2014] Feldman, A., deKleer, J., Narasimhan, S., Poll, S., Garcia, D., and Kuhn, L. (2014). The diagnostic competitions. *AI Magazine*, 35(2) :49–54.

[Feldman et al., 2010a] Feldman, A., Provan, G., and van Gemund, A. (2010a). Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research*, 38 :371–413.

[Feldman et al., 2010b] Feldman, A., Provan, G., and van Gemund, A. (2010b). A model-based active testing approach to sequential diagnosis. *Journal of Artificial Intelligence Research*, 39 :301–304.

[Fessant et al., 2004] Fessant, F., Clérot, F., and Dousson, C. (2004). Mining of an alarm log to improve the discovery of frequent patterns. In *4th Industrial Conference on Data Mining*, pages 144–152, Leipzig, Germany.

[Fleury et al., 1997] Fleury, S., Herrb, M., and Chatila, R. (1997). Genom : A tool for the specification and the implementation of operating modules in a distributed robot architecture. In *International Conference On Intelligent Robots and Systems*, pages 842–848, Grenoble, France.

[Floyd, 1962] Floyd, R. W. (1962). Algorithm 97 : Shortest path. *Communications of the ACM*, 5(6) :345.

[Forbus and deKleer, 1993] Forbus, K. and deKleer, J. (1993). *Building Problem Solvers*. MIT Press.

[Förstner et al., 2002] Förstner, D., Jung, M., and Lunze, J. (2002). A discrete-event model of asynchronous quantised systems. *Automatica*, 38(8) :1277–1286.

[Friedrich et al., 1992] Friedrich, G., Gottlob, G., and Nejdl, W. (1992). Formalizing the repair process. In *10th European Conference on Artificial Intelligence*, pages 709–713, Vienna, Austria.

[Friedrich et al., 1994] Friedrich, G., Gottlob, G., and Nejdl, W. (1994). Formalizing the repair process – extended report. *Annals of Mathematics and Artificial Intelligence*, 11 :187–201.

[Frisk and Nielsen, 2006] Frisk, E. and Nielsen, L. (2006). Robust residual generation for diagnosis including a reference model for residual behavior. *Automatica*, 42(3) :437–445.

[Gamper and Nejdl, 1997] Gamper, J. and Nejdl, W. (1997). Abstract temporal diagnosis in medical domains. *Artificial Intelligence in Medicine*, 10(3) :209–234.

[Gaudel et al., 2015] Gaudel, Q., Chanthery, E., and Ribot, P. (2015). Hybrid particle petri nets for systems health monitoring under uncertainty. *International Journal of Prognostics and Health Management*, 6(22).

[Genesereth, 1984] Genesereth, M. R. (1984). The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24 :411–436.

[Gertler, 1998] Gertler, J. (1998). *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker Inc.

[Ghallab, 1996] Ghallab, M. (1996). On chronicles : Representation, on-line recognition and learning. In *5th International Conference on Principles of Knowledge Representation and Reasoning*, pages 597–606, Cambridge, Massachusetts, United States.

[Ghallab et al., 2016] Ghallab, M., Nau, D., and Traverso, P. (2016). *Automated Planning and Acting*. Cambridge University Press.

[Ghazel and Liu, 2016] Ghazel, M. and Liu, B. (2016). A customizable railway benchmark to deal with fault diagnosis issues in des. In *13th International Workshop on Discrete Event Systems*, pages 177–182, Xi'an, China.

[Ghazel et al., 1999] Ghazel, M., Toguyéni, A., and Yim, P. (1999). State observer for des under partial observation with time petri nets. *Journal of Discrete Event Dynamic Systems : Theory and Applications*, 19(2) :137–165.

[Ghelam et al., 2006] Ghelam, S., Simeu-Abazi, Z., Derain, J.-P., Feuillebois, C., Vallet, S., and Glade, M. (2006). Integration of health monitoring in the avionics maintenance system. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 1519–1524, Beijing, China.

[Ghosh et al., 2007] Ghosh, D., Sharman, R., Rao, H. R., and Upadhyaya, S. (2007). Self-healing systems survey and synthesis. *Decision Support Systems*, 42(4) :2164–2185.

[Giarratano and Riley, 2004] Giarratano, J. and Riley, G. (2004). *Expert Systems : Principles and Programming (4th ed)*. Course Technology.

[Giua, 2007] Giua, A. (2007). A benchmark for diagnosis.

[Goh et al., 2006] Goh, K., Tjahjono, B., Baines, T., and Subramaniam, S. (2006). A review of research in manufacturing prognostics. In *IEEE International Conference on Industrial Informatics*, pages 417–422, Singapore, Singapore.

[Grastien, 2009] Grastien, A. (2009). Symbolic testing of diagnosability. In *20th Inetrnational Workshop on Principles of Diagnosis*, pages 131–138, Stockholm, Sweden.

[Grastien et al., 2007] Grastien, A., Anbulagan, A., and Kelareva, E. (2007). Diagnosis of discrete-event systems using satisfiability algorithms. In *Twenty-Second Conference on Artificial Intelligence*, pages 305–310, Vancouver, Canada.

[Grastien and Cordier, 2007] Grastien, A. and Cordier, M.-O. (2007). Exploiting independence in a decentralised and incremental approach of diagnosis. In *20th International Joint Conference on Artificial Intelligence*, pages 292–297, Hyderabad, India.

[Guan et al., 2011] Guan, X., Liu, Y., Jha, R., Saxena, A., Celaya, J., and Goebel, K. (2011). Comparison of two probabilistic fatigue damage assessment approaches using prognostic performance metrics. *International Journal of Prognostics and Health Management*, 11(5).

[Gucik-Derigny et al., 2009] Gucik-Derigny, D., Outbib, R., and Ouladsine, M. (2009). Estimation of damage behaviour for model-based prognostic. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 1444–1449, Barcelona, Spain.

[Guerraz and Dousson, 2004] Guerraz, B. and Dousson, C. (2004). Chronicles construction starting from the fault model of the system to diagnose. In *15th International Workshop on Principles of Diagnosis*, pages 51–56, Carcassonne, France.

[Guyet and Quiniou, 2011] Guyet, T. and Quiniou, R. (2011). Extracting temporal patterns from interval-based sequences. In *22nd International Joint Conference on Artificial Intelligence*, pages 1306–1311, Barcelona, Spain.

[Haar et al., 2009] Haar, S., Benveniste, A., Fabre, E., and Jard, C. (2009). Partial order diagnosability of discrete event systems using petri net unfolding. In *42nd IEEE Conference on Decision and Control*, pages 3748–3753, Maui, Hawai, United States.

[Haimowitz and Kohane, 1993] Haimowitz, I. J. and Kohane, I. S. (1993). Automated trend detection with alternate temporal hypotheses. In *13th International Joint Conference on Artificial Intelligence*, pages 146–151, Chambéry, France.

[Hamscher, 1988] Hamscher, W. (1988). *Model-based troubleshooting of digital systems*. MIT Press.

[Hamscher et al., 1992] Hamscher, W., Console, L., and de Kleer, J. (1992). *Readings in model-based diagnosis*. Morgan Kaufman.

[Hedman, 2004] Hedman, S. (2004). *A First Course in Logic : An Introduction to Model Theory, Proof Theory, Computability, and Complexity*. Oxford Press.

[Hepes et al., 2009] Hepes, I., Bernard, D., Dekneudt, J., Cheriere, V., and Belard, N. (2009). Comparison of possible approaches of centralized diagnostic. Technical report, Airbus.

[Hodges, 1993] Hodges, W. (1993). *Model Theory*. Cambridge University Press.

[Hopcroft et al., 2001] Hopcroft, J., Motwani, R., and Ullman, J. (2001). *Introduction to automata theory, languages, and computation (2nd ed)*. Addison Wesley.

[Huang and Darwiche, 1996] Huang, C. and Darwiche, A. (1996). Inference in belief networks : A procedural guide. *International Journal of Approximate Reasoning*, 15(3) :255–263.

[Hutchings and Shipway, 2017] Hutchings, I. and Shipway, P. (2017). *Tribology : Friction and Wear of Engineering Materials*. Butterworth-Heinemann.

[Isermann, 2005] Isermann, R. (2005). Model-based fault-detection and diagnosis ? status and applications. *Annual Reviews in Control*, 29 :71–85.

[Isermann and Ballé, 1997] Isermann, R. and Ballé, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice*, 5(5) :709–719.

[Jannach et al., 2016] Jannach, D., Schmitz, T., and Shchekotykhin, K. (2016). Parallel model-based diagnosis on multi-core computers. *Journal of Artificial Intelligence Research*, 55(1) :835–887.

[Jaulin et al., 2001] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. (2001). *Applied Interval Analysis : With Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer.

[Jaulin and Walter, 1993] Jaulin, L. and Walter, E. (1993). Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4) :1053–1064.

[Jensen and Jensen, 1994] Jensen, F. V. and Jensen, F. (1994). Optimal junction trees. In *Tenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 360–366, Seattle, Washington, United States.

[Jéron et al., 2006] Jéron, T., Marchand, H., Pinchinat, S., and Cordier, M.-O. (2006). Supervision patterns in discrete event systems diagnosis. In *8th International Workshop on Discrete Event Systems*, pages 262–268, Ann Arbor, MI, United States.

[Jiang et al., 2001] Jiang, S., Huang, Z., Chandra, V., and Kumar, R. (2001). A polynomial algorithm for testing diagnosability of discrete-event systems. *Transactions on Automatic Control*, 46(8) :1318–1321.

[Jiang and Kumar, 2004] Jiang, S. and Kumar, R. (2004). Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *Transactions on Automatic Control*, 49(6) :934–945.

[Jiroveanu and Boel, 2006] Jiroveanu, G. and Boel, R. K. (2006). A distributed approach for fault detection and diagnosis based on time petri nets. *Mathematics and Computers in Simulation*, 70 :287–313.

[Kan John and Grastien, 2008] Kan John, P. and Grastien, A. (2008). Local consistency and junction tree for diagnosis of discrete-event systems. In *18th European Conference on Artificial Intelligence*, pages 209–213, Patras, Greece.

[Keller, 2007] Keller, K. (2007). Health management technology integration. In *18th International Workshop on Principles of Diagnosis*, Nashville, Tennessee, United States.

[Khoumsi and Ouedraogo, 2009] Khoumsi, A. and Ouedraogo, L. (2009). Diagnosis of faults in real-time discrete event systems. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 1557–1562, Barcelona, Spain.

[Kockskämper et al., 1994] Kockskämper, S., Neumann, B., and Schick, M. (1994). Extending process monitoring by event recognition. In *Second International Conference on Intelligent Systems Engineering*, pages 455–460, Hamburg, Germany.

[Koitz and Wotawa, 2015] Koitz, R. and Wotawa, F. (2015). Sat-based abductive diagnosis. In *26th international workshop on Principles of Diagnosis*, pages 167–176, Paris, France.

[Kothamasu et al., 2006] Kothamasu, R., Huang, S. H., and Verduin, W. (2006). System health monitoring and prognostics a review of current paradigms and practices. *International Journal of Advanced Manufacturing Technology*, 28 :1012–1024.

[Kowalski and Sergot, 1986] Kowalski, R. and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4(1) :67–95.

[Kripke, 1963] Kripke, S. (1963). Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16 :83–94.

[Kuhn et al., 2010] Kuhn, L., Price, B., Do, M., Liu, J., Zhou, R., Schmidt, T., and deKleer, J. (2010). Pervasive diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, 40(5) :932–944.

[Kuznetsov, 2004] Kuznetsov, Y. N. (2004). *Elements of Applied Bifurcation Theory, Third Edition*. Springer.

[Laborie and Krivine, 1997b] Laborie, P. and Krivine, J.-P. (1997b). Automatic generation of chronicles and its application to alarm processing in power distribution systems. In *8th International Workshop on Principles of Diagnosis*, pages 61–68, Mont Saint-Michel, France.

[Lamperti and Zanella, 2002] Lamperti, G. and Zanella, M. (2002). Diagnosis of discrete-event systems from uncertain temporal observations. *Artificial Intelligence*, 137 :91–163.

[Lamperti and Zanella, 2003] Lamperti, G. and Zanella, M. (2003). *Diagnosis of Active Systems : Principles and Techniques*. Springer.

[Lamperti and Zanella, 2006] Lamperti, G. and Zanella, M. (2006). Flexible diagnosis of discrete-event systems by similarity-based reasoning techniques. *Artificial Intelligence*, 170 :232–297.

[Le Guillou et al., 2008] Le Guillou, X., Cordier, M.-O., Robin, S., and Rozé, L. (2008). Chronicles for on-line diagnosis of distributed systems. In *18th European Conference on Artificial Intelligence*, pages 194–198, Patras, Greece.

[Lebold and Thurston, 2006] Lebold, M. and Thurston, M. (2006). Standards developments for condition-based maintenance systems. In *55th Meeting of the Society for Machinery Failure Prevention Technology*, pages 363–373, Virginia Beach, USA, United States.

[Lefebvre and Delherm, 2007] Lefebvre, D. and Delherm, C. (2007). Diagnosis of des with petri net models. *IEEE Transactions on Automation Science and Engineering*, 4(1) :114–118.

[Lin, 1994] Lin, F. (1994). Diagnosability of discrete event systems and its applications. *Journal of Discrete Event Dynamic Systems : Theory and Applications*, 4(2) :197–212.

[Liu et al., 2014a] Liu, B., Ghazel, M., and Toguyéni, A. (2014a). Diagnosis of labeled time petri nets using time interval splitting. In *19th World Congress of The International Federation of Automatic Control*, pages 1784–1789, Cape Town, South Africa.

[Liu et al., 2014c] Liu, B., Ghazel, M., and Toguyéni, A. (2014c). Toward an efficient approach for diagnosability analysis of des modeled by labeled petri nets. In *13th European Control Conference*, pages 1293–1298, Strasbourg, France.

[Luo et al., 2008] Luo, J., Pattipati, K. R., Qiao, L., and Chigusa, S. (2008). Model-based prognostic techniques applied to a suspension system. *IEEE Transactions on Systems, Man and Cybernetics*, 38(5) :1156–1168.

[Mannila et al., 1997] Mannila, H., Toivonen, H., and Verkano, A. I. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1997(1) :249–289.

[Marchand and Rozé, 2002] Marchand, H. and Rozé, L. (2002). Diagnostic de pannes sur des systèmes à événements discrets : une approche à base de modèles symboliques. In *13ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle*, pages 191–200, Angers, France.

[Marker, 2002] Marker, D. (2002). *Model Theory : An Introduction*. Springer.

[Marques-Silva et al., 2015] Marques-Silva, J., Janota, M., Ignatiev, A., and Morgado, A. (2015). Efficient model based diagnosis with maximum satisfiability. In *24th International Joint Conference on Artificial Intelligence*, pages 1966–1972, Buenos Aires, Argentina.

[Martelli and Montanari, 1973] Martelli, A. and Montanari, U. (1973). Additive and/or graphs. In *3rd international joint conference on Artificial intelligence*, pages 1–11, Stanford, United States.

[MaxPlus, 1991] MaxPlus (1991). Second order theory of min-linear systems and its application to discrete event systems. In *30th IEEE Conference on Decision and Control*, Brighton, United Kingdom.

[Mayer, 1998] Mayer, E. (1998). Inductive learning of chronicles. In *13th European Conference on Artificial Intelligence*, pages 471–472, Brighton, United Kingdom.

[Mayer et al., 2012] Mayer, W., Friedrich, G., and Stumptner, M. (2012). On computing correct processes and repairs sing partial behavioral models. In *20th European Conference on Artificial Intelligence*, pages 582–587, Montpellier, France.

[McIlraith, 1995] McIlraith, S. (1995). Incorporating action into diagnostic problem solving (an abridged report). In *AAAI Spring Symposium on Extending Theories of Action : Formal Theory and Practical Applications*, pages 139–144, Stanford, California, United States.

[McMillan and Probst, 1995] McMillan, K. L. and Probst, D. K. (1995). A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1) :45–65.

[Meuleau et al., 2009] Meuleau, N., Bénazéra, E., Brafman, R. I., Hansen, E., and Mausam (2009). A heuristic search approach to planning with continuous resources in stochastic domains. *Journal of Artificial Intelligence Research*, 34 :27–59.

[Mitsa, 2010] Mitsa, T. (2010). *Temporal Data Mining*. CRC Press.

[Moore, 1966] Moore, R. E. (1966). *Interval analysis*. Prentice-Hall.

[Moreira et al., 2011] Moreira, M. V., Jesus, T., and Basilio, J. a. (2011). Polynomial time verification of decentralized diagnosability of discrete event systems. *Transactions on Automatic Control*, 56(7) :1679–1684.

[Morin and Debar, 2003] Morin, B. and Debar, H. (2003). Correlation on intrusion : an application of chronicles. In *6th International Conference on recent Advances in Intrusion Detection*, pages 94–112, Pittsburgh, PA, United States.

[Mozetic and Holzbaur, 1994] Mozetic, I. and Holzbaur, C. (1994). Controlling the complexity in model-based diagnosis. *Annals of Mathematics and Artificial Intelligence*, 11 :297–314.

[Narasimhan et al., 1998] Narasimhan, S., Mosterman, P. J., and Biswas, G. (1998). A systematic analysis of measurement selection algorithms for fault isolation in dynamic systems. In *9th International Workshop on Principles of Diagnosis*, pages 94–101, Cape Cod, Massachusetts, United States.

[Nedialkov, 2006] Nedialkov, N. S. (2006). Vnode-lp a validated solver for initial value problems in ordinary differential equations. Technical Report 606, Department of Computing and Software.

[Nejdl and Bachmayer, 1993] Nejdl, W. and Bachmayer, J. (1993). Diagnosis and repair iteration planning versus n-step look ahead planning. In *4th International Workshop on Principles of Diagnosis*, Aberystwyth, United Kingdom.

[Nokel, 1989] Nokel, K. (1989). Temporal matching : Recognizing dynamic situations from discrete measurements. In *11th international joint conference on Artificial intelligence*, pages 1255–1260, Detroit, MI, United States.

[Olive et al., 2003] Olive, X., Poulard, H., and Travé-Massuyès, L. (2003). Ao* variant methods for automatic generation of near-optimal diagnosis trees. In *14th International Workshop on Principles of Diagnosis*, pages 169–174, Washington, DC, United States.

[Pandalai and Holloway, 2000] Pandalai, D. N. and Holloway, L. E. (2000). Template languages for fault monitoring of timed discrete event processes. *Transactions on Automatic Control*, 45(5) :868–882.

[Pattipati and Dontamsetty, 1992] Pattipati, K. R. and Dontamsetty, M. (1992). On a generalized test sequencing problem. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2) :392–396.

[Patton et al., 2000] Patton, R., Frank, P., and Clark, R. (2000). *Issues of Fault Diagnosis for Dynamic Systems*. Springer.

[Peng and Reggia, 1990] Peng, Y. and Reggia, J. (1990). *Abductive Inference Models for Diagnostic Problem-Solving*. Springer.

[Peot, 1992] Peot, M. A. (1992). Conditional nonlinear planning. In *First International Conference on AI Planning Systems*, pages 189–197, College Park, Maryland, United States.

[Peterson, 1977] Peterson, J. L. (1977). Petri nets. *ACM Computing Surveys*, 9(3) :223–252.

[Philippot et al., 2012] Philippot, A., Tajer, A., and Carré-Ménétrier, V. (2012). From centralized to decentralized approach for optimal controller of discrete manufacturing systems. *ARPN Journal of Science and Technology*, 2(10) :936–949.

[Point and Rauzy, 1999] Point, G. and Rauzy, A. (1999). Altarica - constraint automata as a description language. *European Journal on Automation*, 33 :1033–1052.

[Poole, 1989a] Poole, D. (1989a). Explanation and prediction : An architecture for default and abductive reasoning. *Computational Intelligence*, 5(2) :97–110.

[Pryor and Collins, 1996] Pryor, L. and Collins, G. (1996). Planning for contingencies : A decision-based approach. *Journal of Artificial Intelligence Research*, 4 :287–339.

[Pucel et al., 2007] Pucel, X., Bocconi, S., Picardi, C., Theiseder-Dupré, D., and Travé-Massuyès, L. (2007). Diagnosability analysis for web services with constraint-based models. In *18th International Workshop on Principles of Diagnosis*, pages 360–367, Nashville, Tennessee, United States.

[Py et al., 2006] Py, F., Perrot, F., Pencolé, Y., Orlandini, A., Travé-Massuyès, L., and Félix, I. (2006). Projet agata. rapport de synthèse sur l'activité : études et définitions de modèles. Technical report, LAAS-CNRS.

[Ramadge and Wonham, 1989] Ramadge, P. J. and Wonham, W. (1989). The control of discrete event processes. *Proceedings of the IEEE*, 77(1) :81–98.

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

[Ressencourt et al., 2006] Ressencourt, H., Travé-Massuyès, L., and Thomas, J. (2006). Hierarchical modelling and diagnosis for embedded systems. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, pages 553–558, Beijing, China.

[Rintanen and Grastien, 2007] Rintanen, J. and Grastien, A. (2007). Diagnosability testing with satisfiability algorithms. In *20th International Joint Conference on Artificial Intelligence*, pages 532–537, Hyderabad, India.

[Roemer and Byington, 2007] Roemer, M. and Byington, C. (2007). Prognostics and health management software for gas turbine engine bearings. In *ASME Turbo Expo 2007 : Power for Land, Sea, and Air*, pages 795–802, Montreal, Canada.

[Rossi et al., 2006] Rossi, F., van Beek, P., and Walsh, T. (2006). *Handbook of Constraint Programming*. Elsevier.

[Roth et al., 2011] Roth, M., Lesage, J.-J., and Litz, L. (2011). The concept of residuals for fault localization in discrete event systems. *Control Engineering Practice*, 19(9) :978–988.

[Roychoudhury and Daigle, 2011] Roychoudhury, I. and Daigle, M. J. (2011). An integrated model-based diagnostic and prognostic framework. In *22nd International Workshop on Principle of Diagnosis*, Murnau, Germany.

[Rozé and Cordier, 2002] Rozé, L. and Cordier, M.-O. (2002). Diagnosing discrete-event systems : extending the diagnoser approach to deal with telecommunication networks. *Journal of Discrete Event Dynamic Systems : Theory and Applications*, 12(1) :43–81.

[Sachenbacher and Struss, 2005] Sachenbacher, M. and Struss, P. (2005). Task-dependent qualitative domain abstraction. *Artificial Intelligence*, 162 :121–143.

[Saddem and Philippot, 2014] Saddem, R. and Philippot, A. (2014). Causal temporal signature from diagnoser model for online diagnosis of discrete event systems. In *International Conference on Control, Decision and Information Technologies*, pages 551–556, Metz, France.

[Salaün et al., 2004] Salaün, G., Bordeaux, L., and Schaerf, M. (2004). Describing and reasoning on web services using process algebra. In *International conference on Web Services*, pages 43–51, San Diego, California, United States.

[Sampath et al., 1997] Sampath, M., Lafortune, S., and Teneketzis, D. (1997). Active diagnosis of discrete event systems. In *36th IEEE Conference on Conference : Decision and Control*, pages 2976–2983, San Diego, California, United States.

[Sampath et al., 1995] Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *Transactions on Automatic Control*, 40(9) :1555–1575.

[Sampath et al., 1996] Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1996). Failure diagnosis using discrete event models. *IEEE Transaction on Control Systems Technology*, 4(2) :105–124.

[Schnoebelen, 1999] Schnoebelen, P. (1999). *VERIFICATION DE LOGICIELS. Techniques et outils du model-checking*. Vuibert.

[Schumann and Huang, 2008] Schumann, A. and Huang, J. (2008). A scalable jointree algorithm for diagnosability. In *23rd AAAI Conference on Artificial Intelligence*, pages 535–540, Chicago, United States.

[Schwabacher and Goebel, 2007] Schwabacher, M. and Goebel, K. (2007). A survey of artificial intelligence for prognostics. In *2007 AAAI Fall Symposium Series*, Arlington, Virginia, United States.

[Sengupta, 1998] Sengupta, R. (1998). Diagnosis and communication in distributed systems. In *4th International Workshop on Discrete Event Systems*, pages 144–151, Cagliari, Italy.

[Shaw, 2001] Shaw, M. (2001). Self-healing : softening precision to avoid brittleness : position paper for woss '02 : workshop on self-healing systems. In *First Workshop on Self-Healing Systems*, pages 111–114, Charleston, South Carolina, United States.

[Siddiqi and Huang, 2011] Siddiqi, S. and Huang, J. (2011). Sequential diagnosis by abstraction. *Journal of Artificial Intelligence Research*, 41 :329–365.

[Smith, 2004] Smith, D. E. (2004). Choosing objectives in over-subscription planning. In *14th International Conference on Automated Planning and Scheduling*, pages 393–401, Whistler, British Columbia, Canada.

[Smith and Weld, 1998] Smith, D. E. and Weld, D. S. (1998). Conformant graphplan. In *15th National/10th conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 889–896, Madison, Wisconsin, United States.

[Soldani et al., 2007] Soldani, S., Combacau, M., Subias, A., and Thomas, J. (2007). On-board diagnosis system for intermittent fault : application in automotive industry. In *7th IFAC Conference on Fieldbuses and Networks in Industrial and Embedded Systems*, pages 151–158, Toulouse, France.

[Somenzi, 1998] Somenzi, F. (1998). Cudd : Cu decision diagram package release 2.3.0. Technical report, University of Colorado, Boulder.

[Stern et al., 2017] Stern, R., Kalech, M., Shelly, R., and Feldman, A. (2017). How many diagnoses do we need ? *Artificial Intelligence*, 248 :26–45.

[Struss, 1991] Struss, P. (1991). What's in sd ? towards a theory of modeling for diagnosis. In *2nd International Workshop on Principles of Diagnosis*, pages 41–51, Milan, Italy.

[Su and Wonham, 2005] Su, R. and Wonham, W. (2005). Global and local consistencies in distributed fault diagnosis for discrete-event systems. *Transactions on Automatic Control*, 50(12) :1923–1935.

[Su, 2016] Su, X. (2016). *Time Decomposition for Diagnosis of Discrete Event Systems*. PhD thesis, Australian National University.

[Su and Grastien, 2013] Su, X. and Grastien, A. (2013). Diagnosis of discrete event systems by independent windows. In *The 24th International Workshop on Principles of Diagnosis*, pages 148–153.

[Su and Grastien, 2014] Su, X. and Grastien, A. (2014). Verifying the precision of diagnostic algorithms. In *The 21st European Conference on Artificial Intelligence (ECAI 2014)*, pages 861–866.

[Subias et al., 2014] Subias, A., Travé-Massuyès, L., and Le Corronc, E. (2014). Learning chronicles signing multiple scenario instances. In *19th World Congress of The International Federation of Automatic Control*, pages 10397–10402, Cape Town, South Africa.

[Sun and Weld, 1993] Sun, Y. and Weld, D. S. (1993). A framework for model-based repair. In *11th National Conference on Artificial Intelligence*, pages 182–187, Washington, D.C., United States.

[Sztipanovits and Misra, 1996] Sztipanovits, J. and Misra, A. (1996). Diagnosis of discrete event systems using ordered binary decision diagrams. In *7th International Workshop on Principles of Diagnosis*, Val Morin, Canada.

[Tarski, 1983] Tarski, A. (1983). The concept of truth in formalized languages. In *Logic, Semantics, Metamathematics : papers from 1923 to 1938.*, pages 152–278. Hackett Publishing.

[Team, 2008] Team, C. (2008). choco : an open source java constraint programming library. In *Open-Source Software for Integer and Contraint Programming*, Paris, France.

[Teung, 1994] Teung, R. W. (1994). On noiseless diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, 24(7) :1074–1082.

[Thorsley and Teneketzis, 2007] Thorsley, D. and Teneketzis, D. (2007). Active acquisition of information for diagnosis and supervisory control of discrete event systems. *Journal of Discrete Event Dynamic Systems : Theory and Applications*, 17(4) :531–583.

[Toguyéni et al., 1990] Toguyéni, A., Craye, E., and Gentina, J.-C. (1990). A method of temporal analysis to perform online diagnosis in the context of flexible manufacturing system. In *16th Annual conference of IEEE Industrial Electronics Society*, pages 445–450, Pacific Grove, Californie, United States.

[Torta and Torasso, 2007] Torta, G. and Torasso, P. (2007). Computation of minimal sensor sets from precompiled disciminability relations. In *18th International Workshop on Principles of Diagnosis*, pages 202–209, Nashville, Tennessee, United States.

[Travé-Massuyès et al., 2001] Travé-Massuyès, L., Escobet, T., and Milne, R. (2001). Model-based diagnosability and sensor placement application to a frame 6 gas turbine sub-system. In *17th International Joint Conference on Artificial Intelligence*, pages 551–556, Seattle, Washington, United States.

[Travé-Massuyès et al., 2006] Travé-Massuyès, L., Escobet, T., and Olive, X. (2006). Diagnosability analysis based on component supported analytical redundancy relations. *IEEE Transactions on Systems, Man and Cybernetics, Part A : Systems and Humans*, 36(6) :1146–1160.

[Tripakis, 2002] Tripakis, S. (2002). Fault diagnosis for timed automata. In *7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 205–221, Oldenburg, Germany.

[Vachtsevanos et al., 2006] Vachtsevanos, G., Lewis, F. L., Roemer, M., Hess, A., and Wu, B. (2006). *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*. John Wiley & Sons, Inc.

[Vasquez et al., 2017] Vasquez, J. W., Subias, A., Travé-Massuyès, L., and Jimenez, F. (2017). Alarm management via temporal pattern learning. *Engineering Applications of Artificial Intelligence*, 65 :506–516.

[Vizcarrondo et al., 2013] Vizcarrondo, J., Aguilar, J., Subias, A., and Exposito, E. (2013). Distributed chronicles for recognition of failures in web services composition. In *XXXIX Latin American Computing Conference*, Naiguata, Venezuela, Bolivarian Republic Of.

[Wang et al., 2015] Wang, X., Mahulea, C., and Silva, M. (2015). Diagnosis of time petri nets using fault diagnosis graph. *Transactions on Automatic Control*, 60(9) :2321–2335.

[Warren, 1976] Warren, D. H. (1976). Generating conditional plans and programs. In *2nd Summer Conference on Artificial Intelligence and Simulation of Behaviour*, pages 344–354, Edinburgh, United Kingdom.

[Washington et al., 1999] Washington, R., Golden, K., and Bresina, J. (1999). Plan execution, monitoring and adaptation for planetary rover. In *IJCAI-99 Workshop on Scheduling and Planning meet Real-time Monitoring in a Dynamic and Uncertain World*, Stockholm, Sweden.

[Weld, 1992] Weld, D. S. (1992). Reasoning about model accuracy. *Artificial Intelligence*, 56 :255–300.

[Ye and Dague, 2012] Ye, L. and Dague, P. (2012). A general algorithm for pattern diagnosability of distributed discrete event systems. In *24th IEEE International Conference on Tools with Artificial Intelligence*, pages 130–137, Athens, Greece.

[Yeung and Kwong, 2005] Yeung, D. and Kwong, R. (2005). Fault diagnosis in discrete-event systems : Incomplete models and learning. In *2005 American Control Conference*, pages 3327–3332, Portland, Oregon, United States.

[Yoo and Lafortune, 2002] Yoo, T.-S. and Lafortune, S. (2002). Polynomial-time verification of diagnosability of partially observed discrete-event systems. *Transactions on Automatic Control*, 47(9) :1491–1495.

[Zanella and Lamperti, 2004] Zanella, M. and Lamperti, G. (2004). Diagnosis of discrete-event systems by separation of concerns, knowledge compilation, and reuse. In *16th European Conference on Artificial Intelligence*, pages 838–842, Valencia, Spain.

[Zaytoon and Lafortune, 2013] Zaytoon, J. and Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37 :308–320.

[Zhao and Ouyang, 2006] Zhao, X. and Ouyang, D. (2006). A method of combining se-tree to compute all minimal hitting sets. *Progress in Natural Science*, 16(2) :169–174.

[Zhao and Ouyang, 2007] Zhao, X. and Ouyang, D. (2007). Improved algorithms for deriving all minimal conflict sets in model-based diagnosis. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues. Third International Conference on Intelligent Computing*, pages 157–166, Qingdao, China.

[Zwingelstein, 1995] Zwingelstein, G. (1995). *Diagnostic des défaillances : Théorie et pratique pour les systèmes industriels*. Hermes Science.