Vers le diagnostic embarqué de défaillances dans les systèmes à événements discrets : application au domaine automobile

Siegfried Soldani

THÈSE

en vue de l'obtention du

DOCTORAT DE L'UNIVERSITE DE TOULOUSE
délivré par l'Université Toulouse III-Paul Sabatier

Discipline Systèmes Automatiques

présentée et soutenue

par

Siegfried Soldani

12 Septembre 2008

Vers le diagnostic embarqué de défaillances dans les systèmes à événements discrets : application au domaine automobile.

Jury

Véronique Carré-Ménétrier	*Professeur des universités*	(rapporteur)
Marie-Odile Cordier	*Professeur des universités*	(rapporteur)
Michel Combacau	*Professeur des universités*	(directeur de thèse)
Audine Subias	*Maître de conférences*	(co-encadrante)
Jérôme Thomas	*Ingénieur de Recherche ACTIA*	(co-encadrant)
Denys Bernard	*Ingénieur de Recherche AIRBUS*	(examinateur)

Invités

Christian Desmoulins	*Directeur Général ACTIA*
Louise Travé-Massuyès	*Directrice de Recherche*

École doctorale : Systèmes
Laboratoire d'accueil : Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS
Équipe d'accueil : DIagnostic, Supervision et COnduite des systèmes

Remerciements

LE travail présenté dans ce mémoire a été réalisé au sein du laboratoire commun AUTODIAG, associant la société ACTIA, et les laboratoires du LAAS (Laboratoire d'Analyse et d'Architecture des Systèmes) et de l'IRIT, (Institut de Recherche en Informatique de Toulouse). Je tiens donc à remercier Mr Christian Desmoulins, directeur général d'Actia, ainsi que Mr Malik Ghallab et Mr Raja Chatila, directeurs successifs du LAAS, de m'avoir accueilli au sein de leur société et leur laboratoire pendant toutes ces années de travail.

Je remercie également Mme Louise Travé-Massuyès de m'avoir accueilli au sein du groupe de recherche DISCO (DIagnostic, Supervision et COnduite).

Merci au groupe AUTODIAG (Actia, LAAS et IRIT) pour toutes les remarques et conseils durant nos réunions.

Je remercie vivement les rapporteurs de ce mémoire de thèse, Madame Véronique Carré-Ménétrier, directrice de l'UFR sciences exactes et naturelles de Reims et chercheur au Centre de Recherche en Sciences et Technologies de l'Information et de la Communication (CReSTIC), et Madame Marie-Odile Cordier, Professeur des Universités de Rennes 1 et chercheur à l'Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA). Je leurs suis très reconnaissant pour avoir bien voulu étudier mes travaux et d'avoir apporté un regard critique sur ceux-ci. Merci également à Monsieur Denys Bernard, ingénieur de recherche chez Airbus, d'avoir bien voulu examiner mes travaux, et pour l'intérêt et l'attention qu'il a accordé à cette étude.

Je tiens à remercier Audine Subias, co-encadrante de ma thèse, pour son implication, sa disponibilité, ses conseils, et surtout son amitié. Merci pour tout Audine.

Merci également à Jérome Thomas, mon co-encadrant industriel, pour toutes ces longues discussions, pour ton soutien durant mes doutes, pour ta disponibilité, ton amitié, et pour toutes ces choses que je ne cite pas mais qui ont été si importantes durant ces toutes années de thèse.

Je tiens à exprimer ma profonde reconnaissance à Michel Combacau, mon directeur de thèse. Auprès de toi, j'ai appris énormément. Tu as toujours su me montrer le chemin lorsque j'étais bloqué ; m'encourager dans les moments difficiles ; analyser avec pertinence mes travaux ; être patient et diplomate ; de m'avoir obligé à pousser mes réflexions durant nos discussions ; et pour tout le reste. Merci pour tes qualités scientifiques mais également humaines. Merci pour tout.

Merci à axel et hervé mes deux compères d'Actia, pour toutes les discussions, sérieuses ou non, pour toutes ces soirées jeux, délires, papote,... pour ces sorties plongée, et toute cette bonne humeur que vous mettiez dans le bureau... Tout ceci m'a permis de passer cette thèse dans les meilleurs conditions de travail possibles. Spécial dédicace à fred qui m'a bien aidé dans mon travail de thèse coté applicatif. Merci aussi pour ta culture cinématographique qui nous faisait rêver mais surtout pour ta bonne humeur.

Merci également aux potes du labo : pauline, xavier, françois, mehdi, fabien, vincent, yannick, elodie, nico et les autres. Merci pour toutes ces pauses cafés d'une heure, ces discussions hautement philosophiques mais également pour ces soirées poker et UNO endiablées.

Merci à xavier et perrine, mes collocs qui m'ont supporté cette dernière année, pour tous ces repas, ces soirées inattendues et attendues, ces discussions... (et pour le ménage).

Merci aussi à mes amis de l'ENSIEG : élo, les deux nico, jé, juju, max, et michel. Ces week-ends, ces repas de nouvel an, ces bonnes nouvelles que sont vos mariages et la naissance de tous vos petits bouts de chou, tous ces petits moment m'ont apporté de très grandes joies.

Merci aux EK : Tibo, soso, vinc, gaël, chou et tous les autres. Nos petits we-ends retrouvailles sont toujours de très grand moments de bonheur.

Merci également à pierre, fabien et bruno, mes amis de lycée, pour toutes ces soirées endiablées dans les rues de Paris, Cannes ou Toulouse.

Merci à vous tous pour votre amitié qui permet de supporter les aléas de la vie, les bons comme les mauvais moments.

Grand merci à micka pour ton amitié de toujours.

Je ne saurai également remercier à leur juste valeur ceux qui compte le plus pour moi : mes parents et mon frère. Eux qui m'ont poussé, soutenu, supporté, aidé, donné tout ce qu'ils avaient pour que je réussisse dans mes études et dans ma vie personnelle malgrès toutes les difficultés que vous avez traversées. Cette thèse est l'aboutissement de tous nos efforts et mon cadeau pour tout ce que vous avez fait pour moi. Encore une fois MERCI.

Merci aussi à tous ceux que je ne cite pas mais auxquels je pense très fort ; tous ceux que j'ai rencontrés au cours de mes études et de ma vie, et qui ont été et sont d'un grand soutien.

MERCI

Table des matières

Introduction

Au cours des vingt dernières années, le visage de l'automobile s'est transformé avec une utilisation toujours croissante des fonctions électroniques (allumage électronique, fermeture centralisée...) et de l'informatique embarquée (calculateur d'injection, anti-vol, système anti-blocage, coussins gonflables de sécurité, correcteurs de trajectoire...). Le développement de l'électronique et de l'informatique embarquée se traduit essentiellement par une miniaturisation des composants, une amélioration de leurs performances et une réduction de leurs coûts. Alors que ces évolutions technologiques étaient réservées aux véhicules haut de gamme dans les premières années, aujourd'hui elles touchent l'ensemble des véhicules.

Pour les constructeurs, ces progrès ont été motivés par la mise sur le marché de véhicules offrant de plus en plus de fonctionnalités en terme de sécurité et de confort. Pour faire face à l'augmentation des faisceaux électriques provoquée par l'abondance des fonctionnalités offertes, la technique du multiplexage, technique qui consiste à faire passer plusieurs informations par une seule voie de transmission, a permis de simplifier cette architecture électrique et dans le même temps, d'en augmenter la fiabilité.

Cette complexité grandissante s'accompagne des nouvelles préoccupations : les problèmes de maintenance et de services après-vente. En effet, alors qu'ils n'étaient pas considérés comme prioritaires pour les constructeurs, ils font aujourd'hui partie de leurs axes stratégiques. Les métiers de maintenance automobile sont donc profondément affectés par cette révolution. Alors que dans les années soixante-dix, le garagiste s'armait seulement de sa caisse à outils, aujourd'hui il doit utiliser un outil spécifique qu'il connecte au véhicule pour récupérer toutes les informations nécessaires pour un diagnostic du véhicule. Néanmoins les connaissances du garagiste n'évoluent pas aussi vite que le niveau de complexité du véhicule. C'est pourquoi il est nécessaire de développer de nouvelles méthodes de diagnostic pour réduire cet écart et pouvoir assurer un diagnostic rapide, fiable et précis du véhicule.

Un des problèmes qui se pose aux garagistes est l'apparition de nouvelles défaillances dans les véhicules qui ne sont pas décelables par le garagiste de par leur nature : les défaillances fugitives et intermittentes. Ces défaillances apparaissent et disparaissent lors du fonctionnement du véhicule et ne sont généralement plus présentes lors de la phase de diagnostic en garage. Ces types de défaillances sont très difficiles à diagnostiquer et leurs causes sont très souvent mal connues. Peu de connaissances existent sur ces défaillances dont les causes et les symptômes sont divers. Il faut donc trouver un moyen d'apporter une aide au garagiste pour qu'il puisse avoir une idée de la cause de la défaillance, voire de la localiser.

C'est pourquoi, l'objectif de nos travaux consiste à établir une approche qui permet de

détecter et de localiser, au moins partiellement, ces défaillances fugitives en n'ayant aucune connaissance sur celles-ci. Ne sachant quels systèmes sont à surveiller, nous nous sommes penchés sur les conséquences de ces défauts sur le réseau de communication du véhicule, car c'est sur ce réseau qu'un maximum d'informations circule.

En effet, les systèmes électroniques des véhicules sont composés d'un ensemble de composants interconnectés appelés Unités de Contrôle Électroniques (ou ECU en anglais) qui communiquent entre eux au travers l'échange de messages circulant sur le réseau de communication. C'est le multiplexage. Cette coopération leur permet d'exécuter les fonctions du véhicule telles que la fonction climatisation, les fonctions essuyages, les fonctions ABS, ESP... par l'intermédiaire de capteurs et d'actionneurs. Un défaut sur ces capteurs ou actionneurs peut être mal interprété par l'ECU qui peut envoyer de façon inopinée un message (ou inversement ne pas l'envoyer) sur le réseau. La problématique de nos travaux s'inscrit donc dans la recherche de méthodes pour la détection et la localisation de tels événements.

Aussi, de par la nature discrète du système, nous nous sommes intéressés aux approches de diagnostic à base de modèles dans les systèmes à événements discrets pour finalement adopter l'approche "diagnostiqueur" développée par (Sampath *et al.*, 1998). Cette approche consiste à compiler hors-ligne l'information de diagnostic dans une structure de données (appelé diagnostiqueur) qui relie efficacement les observations aux fautes lors du diagnostic en ligne. Notre approche se différencie dans la mesure où nous n'avons aucune information sur les défauts hormis leurs conséquences sur le réseau de communication. C'est cette information que nous allons utiliser dans les modèles. En outre, dans la plupart des approches, l'événement fautif est considéré comme non observable alors que, dans notre cas, il s'agit d'un événement observable qui correspond, en règle générale, à un événement considéré comme normal. Tous nos travaux reposent donc sur l'utilisation de modèles de bon comportement contrairement aux approches classiques.

Nous présentons, dans le premier chapitre, un état de l'art des différentes approches de diagnostic qui ont un objectif de diagnostic pour les systèmes embarqués. Les domaines sur lesquels nous nous sommes penchés sont les domaines aéronautique et spatial, le domaine automobile et le domaine ferroviaire. Nous pouvons ainsi avoir une vision globale des recherches menées dans la communauté scientifique concernant le diagnostic embarqué. Le chapitre deux est consacré au contexte automobile où il y est décrit l'architecture utilisée dans un véhicule ainsi que les différentes techniques de diagnostic qui existent actuellement chez les garagistes. Nous parlons également dans ce chapitre du projet MODE (Multiple knOwledge Diagnosis Engine) dans lequel s'inscrivent nos travaux. Puis, dans le chapitre trois, nous présentons un état de l'art sur le diagnostic de défaillances dans les systèmes à événements discrets et notamment sur l'approche diagnostiqueur qui se base sur des modèles de défaillances du système, et nous nous positionnons par rapport à ces travaux.

Dans le chapitre quatre, nous introduisons nos travaux en présentant la construction du modèle de bon comportement que nous utilisons, puis nous décrivons la première étape de notre approche : la détection. Celle-ci repose sur le principe de comparaison entre les événements attendus par le modèle de bon comportement et les événements reçus en provenance du système.

Les trois chapitres suivants sont consacrés à l'étape de localisation. Cette étape consiste à

déterminer l'événement qui a pu s'insérer ou celui qui est absent. Dans le chapitre cinq, nous utilisons une fenêtre d'événements sur laquelle nous travaillons pour rétablir la cohérence entre les événements reçus et les trajectoires du modèle. Pour un objectif d'un contexte embarqué, nous avons préféré développer, dans le chapitre six, une nouvelle approche à base de diagnostiqueur utilisant le formalisme des automates. Les problèmes d'explosion du nombre d'états liés au produit synchronisé d'automates nous a conduit à définir une nouvelle approche diagnostiqueur mais en utilisant le formalisme des réseaux de Petri. Cette approche est décrite dans le chapitre sept.

Pour finir, le chapitre huit est dédié à l'application de nos étapes de détection et de localisation sur un exemple issu du domaine des transports collectifs urbains (bus, tramways, métros...). Nous y présentons également les différentes applications qui ont été développées dans cet objectif de surveillance, détection et localisation.

1 Problématique et contexte

Introduction

Les systèmes embarqués sont présents depuis de très nombreuses années dans les domaines de l'aéronautique, l'automobile, l'aérospatial, ou encore le ferroviaire. Au départ, ces systèmes étaient essentiellement mécaniques, hydrauliques et électriques.

Aujourd'hui, avec le développement récent de la micro-électronique, les systèmes embarqués sont repensés. L'électronique complète ou remplace de plus en plus les systèmes mécaniques, hydrauliques... pré-existants et apporte au système de nouvelles possibilités en terme de sécurité, de confort, d'autonomie, et autres. Les systèmes embarqués évoluent et se développent de plus en plus. En outre, les nouvelles normes de sécurité et de pollution ne peuvent pas être appliquées sans l'utilisation de l'électronique, ce qui les rend indispensables.

La fiabilité des systèmes embarqués est par conséquent très importante car ceux-ci, par leurs défaillances peuvent mettre en jeu la vie de personnes, peuvent entraîner des coûts financiers élevés ou peuvent engendrer une défiance systématique de l'utilisateur. C'est pourquoi la détection et le diagnostic de défaillances dans les systèmes embarqués est un enjeu majeur pour les constructeurs. Les systèmes étant de nature très différentes (mécaniques, hydrauliques, électriques, électroniques...), de nombreuses approches ont été étudiées pour résoudre les problèmes de diagnostic dans les systèmes embarqués. Nous nous sommes donc penchés sur les quatre principaux domaines que sont l'automobile, l'aéronautique, l'aérospatial, et le ferroviaire pour avoir une vision globale des différentes approches utilisées dans la littérature pour le diagnostic des systèmes embarqués. Nous nous appuierons sur ces différents domaines pour faire une synthèse des techniques utilisées et des systèmes surveillés, c'est-à-dire déterminer le type de systèmes qui sont étudiés.

1.1 Concepts généraux

Le système est, en règle générale, dans un fonctionnement normal. Lors de l'apparition d'une défaillance, celui-ci passe dans un fonctionnement anormal. Si nous reprenons la définition de (Zwingelstein, 1995), "*une défaillance est une altération ou une cessation de l'aptitude d'un ensemble à accomplir sa ou ses fonctions requises avec les performances définies dans les spécifications techniques*". Ces défaillances peuvent être mineures, majeures, critiques, catastrophiques, ou encore partielles, totales, ou intermittentes. Elles sont détectées par leurs

symptômes.

Identifier le passage du système dans un fonctionnement anormal correspond à détecter un symptôme. Il faut donc surveiller le système, c'est-à-dire recueillir en permanence tous les signaux en provenance du système, reconstituer l'état réel de celui-ci, faire toutes les inférences nécessaires pour produire les données utilisées :

- pour dresser des historiques de fonctionnement,
- le cas échéant, pour mettre en œuvre un processus de traitement de défaillance.

Dans cette définition, la surveillance est limitée aux fonctions qui collectent des informations, les archivent, font des inférences, etc. sans agir réellement sur le système. La surveillance a donc un rôle passif vis-à-vis du système.

La fonction de surveillance a la capacité de détecter les défaillances dont les manifestations sont observables directement ou indirectement. La détection fait partie du mécanisme de surveillance. C'est elle qui permet de déterminer si un système est dans son fonctionnement normal ou dans un fonctionnement anormal. La détection est par conséquent l'étape indispensable pour le traitement d'une défaillance. Quand il n'y a pas de symptômes observables, la détection ne peut se faire et par conséquent le diagnostic ne peut être réalisé. La détection des symptômes d'une défaillance précède la détermination des causes de la défaillance.

Pour cela, un système de surveillance doit s'appuyer sur un ensemble de capteurs placés de façon stratégique et fournissant des indicateurs afin d'obtenir des informations sur le système et son environnement. Cela soulève le problème de l'instrumentation nécessaire au système c'est-à-dire le choix du nombre et du positionnement de capteurs (Nyberg, 2002; Travé-Massuyès *et al.*, 2003).

L'efficacité de la phase de détection repose sur sa capacité à distinguer un comportement normal d'un comportement anormal. Le choix du modèle mis en place dans un but de détection dépend donc des paramètres pertinents et efficaces permettant l'évaluation des comportements observés du système. Ce modèle peut prendre en compte différents aspects (logique, temporel, séquentiel...) selon le type de défaillances visé. Le mécanisme de détection dépend de ce choix. Deux types de modèles sont généralement utilisés pour la détection de comportement anormaux.

- le modèle de comportement normal : la détection est réalisée en comparant les observations provenant du système et celles attendues par le modèle. Si il existe une divergence, alors la détection a lieu.
- le modèle de comportement anormal : la détection est réalisée dans ce cas lorsque les observations correspondent à un modèle de faute inclus dans ce modèle de comportement anormal.

Une fois la défaillance détectée, il s'agit de faire un diagnostic. Cela consiste à établir un lien de cause à effet entre le symptôme observé et la défaillance qui est survenue, ses causes et ses conséquences. Nous considérons donc que la phase de détection est une fonction séparée de la fonction de diagnostic.

Pour le diagnostic, il existe généralement deux étapes :

- l'étape de localisation : elle détermine le sous système fonctionnel à l'origine de la défaillance et progressivement affine cette détermination pour désigner l'organe ou dispositif élémentaire défectueux ;

– l'étape d'identification : elle détermine les modes de fautes qui ont engendré la défaillance constatée.

Les méthodes de diagnostic dépendent de la nature des processus, des systèmes, des composants, du modèle établi, des données disponibles... en d'autres termes, du système à surveiller. Il faudra par conséquent mettre en œuvre à chaque fois des méthodes spécifiques en tenant compte des technologies employées.

1.2 Les approches de diagnostic

Ce paragraphe dresse un état de l'art des méthodes de diagnostic dont le domaine d'étude concerne les systèmes embarqués. Les approches proposées dans la littérature se définissent comme étant des approches de diagnostic embarqué mais peu d'entre elles ont réellement été mises en place sur ce type de système. Néanmoins nous faisons ici une synthèse des différentes approches afin de déterminer celles qui sont les plus pertinentes ainsi que les systèmes sur lesquels elles sont appliquées.

Les systèmes embarqués sont des objets complexes qui s'appuient sur des équipements relevant de plusieurs domaines de la physique (électronique, mécanique, hydraulique, aérodynamique, thermodynamique...). Les approches proposées essayent d'être de plus en plus innovantes et performantes car les méthodes traditionnelles telles que les AMDECs (Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité), les arbres de diagnostic... ne sont plus suffisantes. Les approches de diagnostic diffèrent assez largement selon les connaissances mises en jeu. Trois grandes classes se distinguent : l'approche basée connaissances, l'approche à base de modèles, et l'approche à base de données. La classification (figure 1.1) est reprise de (Venkatasubramanian *et al.*, 2001). Cette classification ne prétend pas être exhaustive mais nous permet de situer les différents travaux et de nous positionner par rapport ceux-ci.

1.2.1 L'approche basée connaissances

L'approche basée connaissances est très peu exploitée pour un diagnostic embarqué et est surtout utilisée comme complément en diagnostic hors-ligne (Murphy et Hershberger, 1996; Bobi *et al.*, 2001; Papadopoulos, 2003). Le principe d'une telle approche est d'associer les causes et les symptômes. Elle peut s'exprimer sous divers formalismes : dictionnaires de pannes, Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité (AMDEC), systèmes à base de règles, arbres de défaillances...

Les AMDEC ont été employées pour la première fois à partir des années 1960 dans le domaine aéronautique pour la sécurité des avions (Recht, 1966). Cette méthode permet une analyse systématique et très complète, composant par composant, de tous les modes de défaillances possibles et de leurs effets sur le système (Villemeur, 1988). La démarche consiste à définir le système, ses fonctions et ses composants. Ensuite, l'ensemble des modes de défaillances des composants doit être établi. L'ensemble des causes susceptibles de provoquer ces défaillances est déterminé. Finalement, les effets sur le système sont étudiés pour chaque mode de défaillance et chaque cause. Le tout est présenté sous forme d'un tableau récapitulatif. Les

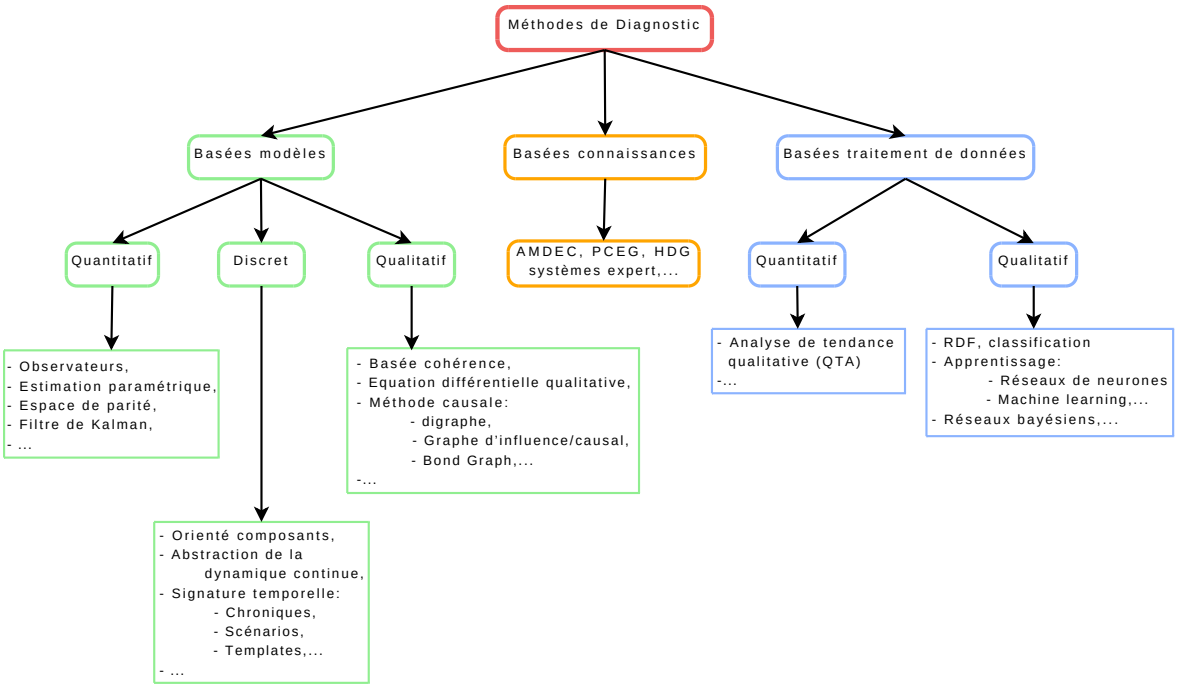


Figure 1.1 — Classification des méthodes de diagnostic

AMDEC sont donc utilisées de façon déductive dans un objectif de diagnostic i.e. que l'on part des effets observés pour remonter aux causes de défaillances possibles (Zwingelstein, 1995). Cette approche est très efficace grâce à ses relations de causes à effets. Néanmoins, les limites d'une telle approche résident dans le fait que les défaillances doivent être définies a priori. Le recensement préalable de ces défaillances ne peut pas être exhaustif et nécessite une longue expérience. Le coût de construction des AMDEC est donc très élevé car cette construction peut être longue et très difficile à mettre en place. En outre, toute évolution ou modification du système implique une réécriture du tableau.

Un arbre de défaillances est une méthode qui permet de déterminer les chemins critiques dans un système (Villemeur, 1988). Elle définit les différentes combinaisons possibles d'événements qui entrainent l'apparition d'un événement non désirable et unique. Cette approche est représentée par une structure arborescente (un arbre) dont le sommet est l'événement indésirable. Il est établi sous forme d'un diagramme logique. Les causes immédiates qui produisent l'événement indésirable sont hiérarchisées par des symboles ET et OU logiques. De cette façon, l'arbre est créé au fur et à mesure pour atteindre des événements dits élémentaires. Le problème de ce type d'approche est leur forte dépendance aux erreurs du concepteur. De plus, comme pour les AMDEC, les défaillances doivent être déterminées a priori et l'évolution du système nécessite une reconstruction de l'arbre de défaillance.

Dans les systèmes à base de règles nous retrouvons les systèmes experts (Buchanan et Shortliffe, 1984). Ce sont des approches qui consistent à reproduire les connaissances et le raisonnement d'un expert dans l'accomplissement d'une tâche. La connaissance utilisée dans ces systèmes est représentée par un ensemble de règles qui sont enchaînées pour simuler le raisonnement de l'expert. Ces règles décrivent les relations fonctionnelles entre les différents composants et les relations causales entre les défaillances et leurs effets. Un moteur d'inférence

est ensuite utilisé pour établir un diagnostic à partir des connaissances contenues dans la base de connaissances. Nous retrouvons deux méthodes de raisonnement. Le chaînage avant qui, à partir d'une déviation, permet d'activer certaines règles dont les conséquences deviennent de nouveaux faits qui activent de nouvelles règles. Le processus s'arrête lorsqu'il n'y a plus de règles à activer. Le chaînage arrière quant à lui permet, à partir d'un fait, de rechercher toutes les règles dont il est la conclusion. Les antécédents de ces règles deviennent des faits à partir desquels le raisonnement se poursuit. Dans le domaine automobile, ce système a été largement utilisé dans le but d'un diagnostic plus systématique et plus précis car en plus de localiser les composants défaillants et leurs causes, le système propose des actions pertinentes pour corriger les défaillances (Gelgele et Wang, 1998). La difficulté des systèmes experts est de pouvoir formaliser sous forme de règles toutes les propriétés d'un système et par conséquent de l'expérience que nous pouvons en avoir (Papadopoulos et MacDermid, 2001). De plus, comme pour les autres approches, les systèmes experts sont basés sur la connaissance que nous avons des défaillances du système. Ils restent également très dépendants du système et par conséquent nécessitent la réécriture des règles et la ré-acquisition de la connaissance des dysfonctionnements du système (Davis et Hamscher, 1988).

Nous avons ici fait une rapide synthèse sur les méthodes à base de connaissances. Ces méthodes ne font pas l'objet d'étude sur les systèmes embarqués. Leur forte dépendance aux systèmes, qui lorsqu'il s'agit de systèmes embarqués peuvent évoluer régulièrement, rend difficile leur utilisation. Nous pouvons remarquer que ces approches nécessitent une connaissance préalable de l'ensemble des défaillances qui peuvent survenir sur le système mais c'est aussi ce qui fait qu'elles sont très efficaces. Néanmoins ces approches sont surtout utilisées pour un diagnostic hors-ligne, après détection d'un symptôme, afin de localiser les composants défaillants et la nature de leur défaillance. Elles sont peu adaptées pour la détection et la localisation en ligne des défaillances.

1.2.2 L'approche à base de modèles

Les approches à base de modèles dans un objectif de diagnostic embarqué sont très largement répandues dans la littérature scientifique. L'approche à base de modèles exploite un modèle du fonctionnement explicite du système à diagnostiquer. Elle fonde la détection des défaillances sur la constatation de discordances entre le comportement prédit par le modèle et le comportement réellement observé, et la localisation sur le recoupement des groupes de composants impliqués dans chacune de ces discordances.

Selon la connaissance que nous avons du système, il est possible de définir deux approches différentes : l'approche FDI (Fault Detection and Isolation) issue de la communauté automatique et dont le principe repose sur des modèles quantitatifs, et l'approche DX issue de la communauté de l'intelligence artificielle et dont le principe repose sur des modèles qualitatifs.

1.2.2.1 L'approche DX

La méthode, dite de diagnostic à base de modèles, ou encore diagnostic à partir des principes premiers, a vu le jour au milieu des années 70 et a été formalisée au début des années 80. Un nombre croissant de travaux a été mené depuis et cette problématique est devenue

un domaine de recherches à part entière de l'I.A. Les articles les plus marquants jusqu'à 1991 sont regroupés dans (de Kleer et Williams, 1989; de Kleer *et al.*, 1992), (Hamscher *et al.*, 1992) ou encore dans (Console et Friedrich, 1994). Une synthèse de ce domaine est faite par P. Dague dans le chapitre 1 de l'ouvrage de (Dubuisson, 2001).

Les connaissances incluses dans les modèles utilisés décrivent la structure (connexions entre composants) du système à diagnostiquer et son comportement (en termes du comportement des différents composants intervenant dans le système). Seules les connaissances structurelles sont spécifiques au système en question ; les connaissances comportementales, idéalement issues des lois de la physique, sont génériques et ne dépendent que du domaine (électronique, thermodynamique, hydraulique, mécanique, etc.).

Ce qui distingue cette approche est qu'il n'est pas nécessaire de raisonner sur des modèles de fautes mais sur les modèles de bon comportement pour déterminer si le système sort de celui-ci. Plus besoin d'anticiper les défauts ou dysfonctionnements pouvant affecter un système et leurs effets pour pouvoir les détecter et les localiser : modéliser le comportement correct est suffisant. L'idée fondamentale est de comparer le comportement réel du système tel qu'il peut être observé par l'intermédiaire des capteurs et son comportement attendu tel qu'il est prédit grâce aux modèles de bon comportement. Toute incohérence entre les observations et les prédictions déduites des modèles est interprétée comme la manifestation d'un dysfonctionnement, i.e. la présence d'un ou plusieurs défauts. Ceci est bien sûr conditionné par le fait que les modèles soient corrects, c'est à dire qu'ils représentent réellement le comportement du système.

Conflits et diagnostics La présence de défauts est détectée par l'observation d'une incohérence entre les observations et les prédictions. Cette incohérence met en évidence qu'un composant ou plusieurs, utilisés pour faire la prédiction, ne suivent plus le modèle de bon comportement. L'un d'entre eux est nécessairement défectueux. L'ensemble de ces composants forme un conflit. La localisation se fait en recoupant les différents conflits obtenus à partir des observations sur le système et des prédictions faites à partir des modèles.

Ensuite, l'objectif est de rétablir la cohérence en faisant des hypothèses sur le fonctionnement des composants. Un diagnostic est l'ensemble des composants qui supposés en fautes permettent de rétablir la cohérence avec les observations. C'est pourquoi le terme de diagnostic à base de cohérence est également employé.

Finalement, lorsqu'il y a plusieurs diagnostics, un raffinement des candidats pourra être utilisé.

Modèles de fautes Dans certains cas, la connaissance de modèles de fautes permet d'enrichir le diagnostic dans la mesure où elle nous permet d'aller plus loin que la simple localisation c'est-à-dire l'identification. Aussi, des extensions du formalisme furent proposées pour permettre d'utiliser des modèles de faute : GDE+ (Struss et Dressler, 1989) et Sherlock (de Kleer et Williams, 1989) sont les travaux les plus marquants.

Aux deux modes de comportement correct et incorrect utilisés précédemment s'ajoutent les modèles des modes de fautes connus, considérés deux à deux exclusifs, ainsi qu'un mode Inconnu sans modèle associé qui représente tous les modes de faute non anticipés.

Ainsi un conflit représente une incohérence entre les observations et les prédictions faites à partir des assignations de modes comportementaux à certains composants. Un diagnostic est le rétablissement de la cohérence par assignation de modes comportementaux à tous les composants du système. La génération des diagnostics étant un problème NP-difficile, diverses techniques de localisation furent proposées (de Kleer, 1991) pour ne générer que les diagnostics préférés selon certains critères (diagnostics les plus probables (de Kleer, 1990), diagnostics explicatifs (Console et Torasso, 1990; Console et Torasso, 1991; de Kleer *et al.*, 1992). Nous retrouvons ces travaux, dans un objectif embarqué, dans (Steinbauer et Wotawa, 2005) par exemple.

1.2.2.2 L'approche FDI

L'application de l'automatique dans un objectif de détection et de diagnostic est apparue dans les années 70 (Rosembrock, 1970). Plusieurs articles de synthèse sont disponibles, notamment (Gertler, 1998), (Isermann, 1997), et (Dubuisson, 2001).

Les modèles sont construits à partir des lois fondamentales (physiques...), et sont décrits par des modèles analytiques. Le principe de détection consiste en la génération d'indicateurs de défauts, appelés résidus. Les systèmes embarqués font appel à des modèles aussi bien linéaires que non linéaires.

La surveillance comprend deux phases :
- le calcul des résidus
- l'évaluation des résidus

Les résidus représentent la différence entre le comportement réel du système et celui prédit par le modèle. Dans le cas nominal, la valeur du résidu est nulle et non nulle lors de la présence d'une défaillance. L'expression des résidus est obtenue a priori, comme une fonction des grandeurs observables (phase de calcul des résidus). Elle est ensuite évaluée en utilisant les mesures (phase d'évaluation des résidus). Une procédure de décision, prenant en compte les caractéristiques statistiques des bruits de mesure et les erreurs de modélisation est en général nécessaire pour évaluer les résidus, c'est-à-dire décider si l'écart par rapport à zéro est significatif.

La génération des résidus à partir des modèles analytiques fait l'objet de nombreuses études. Deux grandes approches se distinguent. La première consiste à surveiller les variables utilisées dans la description du système qui peuvent évoluer au cours du temps. Il s'agit des approches basées sur l'estimation des sorties et celle de l'espace de parité. L'autre approche consiste à estimer les paramètres structuraux du système. Ces paramètres sont généralement constants. Par exemple, dans les systèmes embarqués, les techniques de filtrage de Kalman sont largement employées. Le filtre de Kalman est un filtre récursif qui estime les paramètres structuraux ou de sorties d'un système à partir de mesures incomplètes et bruitées comme c'est généralement le cas dans les systèmes embarqués. Nous pouvons citer les travaux de (Li et Goodall, 2003; Li et Goodall, 2004; Mirabadi *et al.*, 1998) pour le domaine ferroviaire, (Washington, 2000; de Freitas, 2002; Hutter et Dearden, 2003) pour le domaine aéronautique.

Une autre approche utilisant le filtre de Kalman est appelée "interacting multiple models". Cette technique consiste à utiliser un banc de filtres de Kalman où chaque filtre représente

un mode de fonctionnement du système. Cela permet d'estimer l'état du système sous une réduction significative du bruit. Une probabilité pour chaque modèle est calculée pour indiquer le mode courant (Zhang et Li, 1997; Roumeliotis *et al.*, 1998; Hwang *et al.*, 2003).

Le filtre de Kalman s'applique pour des systèmes linéaires. Or les systèmes actuels ne sont pas linéaires. Une variante du filtre de Kalman a été développée, appelée filtre de Kalman étendu, pour les systèmes non linéaires mais linéarisables localement. Nous retrouvons ce genre d'approche dans les systèmes aéronautiques (Chingiz et Fikret, 2005).

D'autres approches et notamment le filtrage particulaire ont été mises en place pour les systèmes non linéaires et non linéarisables. Elles sont basées sur des simulations et des techniques statistiques. Ces travaux sont repris dans le domaine ferroviaire (Li et Goodall, 2004; Li *et al.*, 2007) et le domaine aéronautique (Verma *et al.*, 2004; de Freitas, 2002; Hutter et Dearden, 2003).

D'autres techniques, pouvant utiliser les filtres de Kalman, reconstruisent l'état du système représenté sous forme de représentation d'états. Il s'agit des observateurs d'états. Les travaux reposant sur des observateurs sont publiés dans (Venkateswaran *et al.*, 2002; Jensen et Wisniewski, 2002; Steinbauer et Wotawa, 2005; Kashi *et al.*, 2006) pour les domaines aéronautiques, et dans (Chang *et al.*, 2000; Straky *et al.*, 2002; Shraim *et al.*, 2006) pour le domaine automobile. Lorsqu'il existe plusieurs modèles (de fautes et de bon fonctionnement), une fusion floue des observateurs peut être appliquée comme dans (Patton *et al.*, 1998) pour le domaine ferroviaire. Nous retrouvons également un observateur basé sur l'apprentissage itératif pour les systèmes non linéaires comme les moteurs automobiles (Chen et Saif, 2003).

Pour la localisation, différents types de résidus sont générés : les résidus structurés et les résidus directionnels.

Les résidus structurés sont conçus pour que chaque résidu soit sensible à un sous-ensemble de fautes et insensible aux autres. Aussi, quand une faute survient, certains résidus répondent et d'autres restent à zéro : ceci constitue la signature de la faute. L'ensemble des signatures prédéfinies pour les différentes fautes à considérer constitue la matrice de signatures.

Les résidus directionnels sont conçus de telle sorte que, lorsqu'une faute survient, le vecteur de résidus soit confiné suivant une direction particulière de l'espace des résidus.

Notons qu'en pratique, l'objectif est de structurer ou d'orienter au mieux un ensemble de résidus en minimisant/maximisant la sensibilité des résidus par rapport à divers sous-ensembles de fautes. Par ailleurs, il est préférable de rendre les signatures de faute insensibles aux perturbations. Ainsi, cela revient à résoudre un problème d'optimisation global de manière à construire des résidus sensibles aux défaillances de manière structurée, et robustes vis-à-vis des perturbations et incertitudes du modèle.

Il se peut également que les résidus soient corrélés entre eux. Pour remédier à ce problème, la méthode du maximum de vraisemblance généralisée est usuellement employée. Il s'agit d'une technique qui, sous l'hypothèse que les variables ont une distribution connue, permet d'estimer les paramètres d'un modèle (d'une équation ou d'un système, linéaire ou non) avec des restrictions sur des paramètres (cœfficients, matrices de variances ou covariances) ou non. Cette technique est également utilisée dans le domaine aéronautique (Gomez *et al.*, 2000; Wu et Campion, 2004).

Le point fort de cette approche, qui la distingue des autres, est qu'elle peut accomplir la détection et, dans une large mesure, la localisation de fautes en se fondant uniquement sur un modèle de bon fonctionnement du système, à l'exclusion de toute connaissance préalable sur les fautes ou les modes de défaillances. Les modèles de fautes ne sont pas nécessaires (pour le diagnostic purement fondé sur la cohérence, qui permet la localisation de fautes) mais ceux qui existent peuvent être efficacement pris en compte (extension au diagnostic abductif, qui permet l'identification des fautes et l'explication des symptômes).

1.2.2.3 Conclusion

Le diagnostic à base de modèle pour les systèmes embarqués est très largement étudié dans la communauté de diagnostic. Nous n'avons pas voulu établir une liste exhaustive de toutes les approches qui existent mais plutôt d'en dégager le maximum d'information concernant le diagnostic à base de modèle pour les systèmes embarqués. Les approches à partir de filtre de Kalman, d'observateurs d'état et d'estimation de paramètres sont les plus couramment étudiées. Les approches proposées apportent de très bon résultats mais nécessitent la connaissance du modèle du système à surveiller. Ce système correspond généralement à un ensemble particulier comme les systèmes de suspension, les pneus, les amortisseurs, les moteurs, capteurs, actionneurs... La connaissance du système est très précise. Elle permet de connaître par avance les défaillances qui peuvent apparaître sur ces systèmes. L'ensemble des approches considèrent comme acquis la connaissance des fautes. Néanmoins, l'approche à base de modèle nous permet de faire au moins de la détection si aucune connaissance sur les fautes est établie et notamment les approches par estimation paramétrique permettent de déterminer les variables du système qui sont en fautes.

1.2.3 L'approche à base de données

La complexité que l'on retrouve dans les systèmes surveillés que ce soit en aéronautique, dans le ferroviaire ou l'automobile, rend difficile l'obtention de modèles. Il a donc été développé des techniques ne faisant pas appel à la connaissance des modèles mais se basant sur des données précédemment recueillies.

L'approche à base de données procède par apprentissage numérique et classification en exploitant les données existantes, à l'exclusion de toute forme de modélisation (analytique, symbolique ou autre) et peut aussi se fonder sur des techniques d'apprentissage et de classification symboliques (apprentissage par similarité).

Ces approches permettent d'associer un ensemble de mesures (continues ou discrètes) effectuées sur le système à des états de fonctionnement connus. Elles permettent d'avoir une relation d'un espace caractéristique vers un espace de décision, de façon à minimiser le risque de mauvaise classification. Une première technique est une technique classique de discrimination basée sur les outils de la probabilité. Cette technique peut se montrer insuffisante car elle suppose une connaissance a priori de tous les états de fonctionnement et ne prend pas en compte l'évolution du système. D'autres techniques de discrimination reposent sur l'intelligence artificielle. Ces techniques ont l'avantage de ne pas se baser sur les connaissances a priori des états de fonctionnement mais plutôt sur une phase d'apprentissage. Les techniques

de reconnaissance des formes par réseaux de neurones (Bishop, 1995; Ripley, 1996) et de reconnaissance des formes par la logique floue (Zadeh, 1965; Takagi et Sugeno, 1985; Yager et Filev, 1994) sont celles qui sont les plus utilisées.

Les réseaux de neurones sont des outils de l'intelligence artificielle, capables d'effectuer entre autres des opérations de classification. Leur fonctionnement est inspiré par les principes de fonctionnement des neurones biologiques. Leur principal avantage par rapport aux autres outils est leur capacité d'apprentissage et de généralisation de leurs connaissances à des entrées inconnues. Le processus d'apprentissage est une phase très importante pour qu'une classification puisse se faire avec succès. Plusieurs types de réseaux de neurones et plusieurs algorithmes d'apprentissage existent dans la littérature. Une des qualités de ce type d'outil est son adéquation pour la mise au point de systèmes de surveillance modernes, capables de s'adapter à un système complexe avec reconfigurations multiples. L'expert humain joue un rôle très important dans ce type d'application. Toute la phase d'apprentissage supervisé du réseau de neurones dépend de son analyse des modes de fonctionnement du système. Chaque mode doit être caractérisé par un ensemble de données recueillies sur le système. A chaque mode est associé une expertise faite par l'expert. Cette association (ensemble de données - modes de fonctionnement) est apprise par le réseau de neurones. Après cette phase d'apprentissage, le réseau de neurones associe les classes représentant les modes de fonctionnement aux formes d'entrée caractérisées par les données du système. Pour les systèmes embarqués, nous pouvons citer les travaux de (Sharkey *et al.*, 2000; Bobi *et al.*, 2001) pour le domaine ferroviaire, (Chen et Lee, 2002) dans l'aéronautique, ou (Poulard, 1996; Jakubek et Strasser, 2002) pour le domaine automobile.

En plus de leur utilisation pour l'estimation de classes, les réseaux de neurones sont également largement utilisés pour l'estimation de paramètres. L'objectif est d'établir une estimation de paramètres afin de voir s'ils correspondent au bon fonctionnement du système (valeurs seuils...). De nombreux travaux se basant sur ce type d'approche sont publiés, et notamment dans (Debiolles *et al.*, 2004; A.Debiolles *et al.*, 2006) pour le domaine ferroviaire, (Chen et Lee, 2002; Al-Malki et Gu, 2003; Fekih *et al.*, 2006; Al-Malki et Gu, 2006; Fellouah *et al.*, 2006) dans l'aéronautique, ou encore (Schwarte et Isermann, 2002; Kimmich *et al.*, 2005; Nitsche *et al.*, 2004; Capriglione *et al.*, 2003; Capriglione *et al.*, 2007) pour le domaine automobile.

En reconnaissance des formes par approche floue, les classes sont représentées par des sous-ensembles flous. Une fonction dite d'appartenance quantifie le degré d'appartenance entre les mesures et les classes. La mise en œuvre d'une méthode de classification floue implique deux étapes : la construction des fonctions d'appartenance et la définition des règles de décision. Dans les systèmes embarqués, nous pouvons citer les travaux de (Skarlatos *et al.*, 2004) pour le domaine ferroviaire, (Ganguli, 2003; Yu *et al.*, 2004) pour le domaine aéronautique et (Murphey, 2002; Boatas *et al.*, 2000; Boatas, 2001; Crossman *et al.*, 2003; Lu *et al.*, 2003) pour le domaine automobile.

1.2.3.1 Conclusion

Là encore, nous n'avons pas voulu établir une liste exhaustive de toutes les approches à base de données pour le diagnostic embarqué. Nous avons essentiellement voulu avoir un

aperçu général des méthodes employées. Notons que les réseaux de neurones restent l'approche la plus employée même si d'autres approches comme la reconnaissance de forme par logique floue est également très utilisée. L'avantage de ces approches reste l'absence de modèle du système qui n'est pas forcément connu. Elles ne nécessitent aucune connaissance du modèle du système. Les réseaux de neurones sont utilisés dans beaucoup de domaines et d'approches, notamment pour l'estimation de paramètres, l'estimation de classes... Les systèmes considérés restent identiques à ceux des approches basées modèles c'est-à-dire moteurs, capteurs, actionneurs... Là encore, les auteurs ont connaissance des défaillances qui peuvent survenir dans le système. Néanmoins, ces approches permettent de détecter des défaillances sans en avoir une quelconque connaissance. Dans ce cas, lors de la phase d'apprentissage, le modèle est obtenu uniquement à partir des données de bon fonctionnement. Ce modèle sert de référence, ce qui permet à ces approches de déterminer par la suite si le système est dans le bon fonctionnement ou non. Cependant, le système surveillé reste limité à un système particulier.

1.3 Conclusion

L'objectif de ce chapitre était de faire un tour d'horizon des approches de diagnostic dans les systèmes embarqués et plus spécifiquement dans les domaines de l'aéronautique, l'aérospatial, le ferroviaire, et l'automobile. De nombreuses études ont été menées dans ces domaines et il est difficile d'en faire une synthèse exhaustive. Cependant, nous avons voulu ici établir une liste des approches les plus couramment utilisées dans ces différents domaines. Nous pouvons néanmoins remarquer que les études menées traitent de systèmes assez variés mais que ceux-ci restent spécifiques. En effet, les systèmes qui sont étudiés sont, la plupart du temps, des systèmes dont le comportement est connu, tout comme les défaillances qui peuvent s'y produire. Or, actuellement, l'industrie automobile connait de profonds changements qui impliquent des modifications dans l'architecture du véhicule, dans les composants utilisés, dans les fonctions proposées... Les systèmes retrouvés dans un véhicule sont perpétuellement en train de se renouveler dû en grande partie aux progrès de l'électronique et de son utilisation croissante dans les systèmes embarqués du véhicule. Or les approches développées sont très dépendantes des systèmes étudiés et doivent donc être modifiées à chaque fois que le système change. En outre, les défaillances considérées sont propres au système et les techniques développées sont propres à ces défaillances. Or qu'en est-il des systèmes comme une automobile, où il peut exister des défaillances dont l'origine est inconnue ? Quels sont les systèmes qui doivent être surveillés ?

Les approches sont nombreuses et celles à base de modèles restent les plus couramment utilisées. Elles nécessitent par définition un modèle dont l'obtention n'est pas évidente pour plusieurs raisons :

1. les modèles peuvent être difficiles à obtenir dans le sens où le système surveillé est complexe à modéliser ou peut être trop coûteux ;
2. les modèles des systèmes automobiles appartiennent aux constructeurs et il est difficile de les obtenir ;
3. les modèles ne correspondent plus à l'implémentation faite dans le véhicule (évolution

des modèles, différence entre modèle et implémentation...) ;

4. les modèles de fautes dépendent des connaissances que nous avons du système surveillé ;
5. les fautes sont également difficiles à modéliser.

Les approches basées modèles sont très performantes car elles permettent de surveiller le système et de suivre le bon comportement de celui-ci. La connaissance des défaillances dans un système et mieux encore le modèle de celles-ci est un atout certain. Malheureusement, la progression de l'électronique dans les véhicules a fait apparaître de nombreuses défaillances inconnues. Si une telle connaissance existe alors les approches basées modèles sont celles qui donnent les meilleurs résultats. Cependant, ces approches peuvent être utilisées pour la surveillance ou l'estimation de paramètres. Cela ne nécessite aucune connaissance sur les défaillances.

En l'absence de modèle, les réseaux de neurones semblent donner les meilleurs résultats même si leur phase d'apprentissage reste délicate. Mais là encore, les approches à base de données sont utilisées pour classer le système dans un certain mode de fonctionnement et notamment de fautes, ce qui présuppose la connaissance des défaillances qui peuvent survenir sur le système. Néanmoins, ces approches peuvent servir également pour la détection et avoir un mode inconnu par exemple, ou bien dans l'estimation de paramètres.

Pour conclure, la plupart des approches portent sur un système spécifique, connu, et cherchent à détecter et à localiser des défaillances également connues sur le système. Finalement, l'approche à employer dépend de la connaissance ou non du modèle, de sa précision, de la connaissance des fautes sur le système ou encore de la possibilité d'effectuer la phase d'apprentissage.

2 Contexte automobile

Introduction

Ce second chapitre a pour objectif de définir les caractéristiques du domaine automobile, et notamment son évolution en terme d'architecture de système, en terme de pannes mais également en terme de diagnostic. Nous présenterons les problèmes liés à ces évolutions afin d'en extraire les principaux objectifs d'un point de vue diagnostic pour l'automobile d'aujourd'hui.

Après avoir décrit les principales évolutions des systèmes traités, nous étudierons le processus de diagnostic réalisé au niveau du garage. Puis, nous présenterons les solutions de diagnostic envisagées pour la résolution de ces problèmes. Ces solutions s'intègreront dans un système de diagnostic appelé MODE (Multiple knOwledge Diagnosis Engine) auxquels nos travaux apportent une pierre.

2.1 Évolution des véhicules

L'industrie automobile a subi ces dernières années une véritable révolution avec l'introduction massive des fonctions électroniques. Alors que ces évolutions technologiques étaient, au début, réservées aux véhicules haut de gamme, cette révolution touche maintenant l'ensemble des véhicules.

Pour le constructeur, cette évolution des technologies a été motivée par la mise sur le marché de véhicules offrant des fonctionnalités de plus en plus sophistiquées en terme de sécurité, de confort et autres. En outre, les normes actuelles de sécurité et de pollution sont très contraignantes et il impossible aux constructeurs de les satisfaire sans utiliser l'électronique. Ces systèmes électroniques sont appliqués à différents domaines comme :

- le contrôle moteur : injection de carburant, allumage électronique, régulation de ralenti, anti-pollution...
- le comportement routier : anti-bloquage de roues (ABS), suspensions actives, correcteurs de trajectoires (ESP), anti-patinage,...
- la sécurité active : coussins gonflables de sécurité (airbag)...
- le confort : climatisation, chauffage, régulation de vitesse, fermeture centralisée, anti-vol...

Les systèmes automobiles font aujourd'hui partie des systèmes appelés "mécatroniques". Un

système mécatronique est la combinaison synergique et systémique de la mécanique, de l'électronique et de l'informatique temps réel. L'intérêt de ce domaine d'ingénierie interdisciplinaire est de concevoir des systèmes automatiques puissants et de permettre le contrôle de systèmes hybrides complexes.

L'ingénierie de tels systèmes mécatroniques nécessite la conception simultanée et pluridisciplinaire de trois sous-systèmes (cf. figure 2.1) :

- une partie interface Homme/Machine (forme géométrique et dialogue du système à dominante ergonomique et esthétique),
- une partie commande (intelligence embarquée du système à dominante électronique et informatique temps réel),
- une partie opérative (squelette et muscle du système à dominante mécanique et électromécanique).

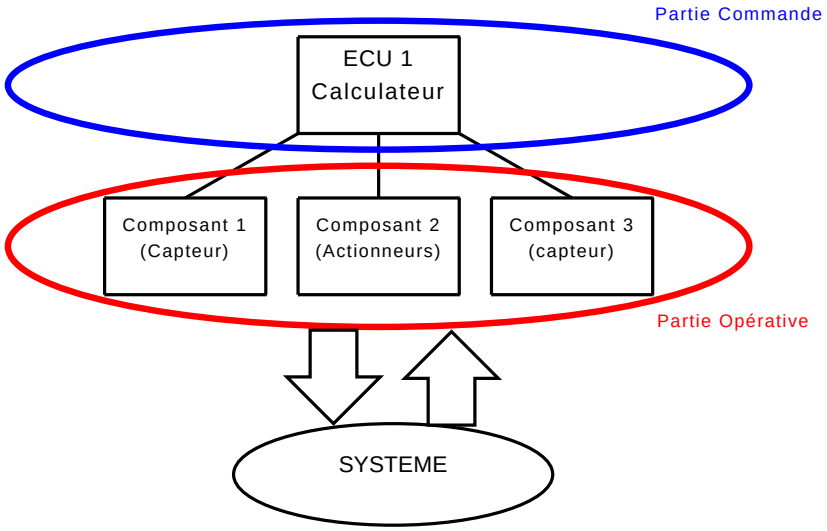


Figure 2.1 — Système mécatronique

2.1.1 La partie commande

La partie commande est constituée essentiellement de calculateurs (ou Electronic Control Units (ECU)). La fonction principale d'un calculateur est d'élaborer les signaux de commande des actionneurs à partir des signaux envoyés par les capteurs. Les calculateurs intègrent des fonctions complexes régissant les lois de commande du système afin de réaliser les fonctionnalités (en terme de confort, sécurité...) exigées par les constructeurs.

Les calculateurs sont aussi dotés d'une fonction d'autodiagnostic dont le but est de détecter, de mémoriser, de signaler et de récupérer des défauts de fonctionnement. La détection de ces défauts est réalisée par des moyens matériels et logiciels. Cela consiste à détecter des courts-circuits ou des circuits ouverts à ses bornes pour la détection matérielle ou alors à détecter des incohérences par dépassement de seuils ou par une incompatibilité de la valeur avec le mode de fonctionnement courant pour la détection logicielle. Ces défauts sont ensuite mémorisés et signalés par un voyant sur le tableau de bord pour les défauts les plus graves. Ils sont également récupérés par le garagiste par l'intermédiaire d'un outil de diagnostic. Ce

sont les codes défauts du véhicule.

2.1.2 La partie opérative

La partie opérative est principalement constituée par les actionneurs et les capteurs.

2.1.2.1 Les capteurs

Le rôle des capteurs est de fournir un signal électrique qui est l'image d'un paramètre physique du système. En ce qui nous concerne, les signaux électriques peuvent être logiques, analogiques, périodiques ou apériodiques. Typiquement, les capteurs qui sont utilisés dans le domaine automobile sont :

- des capteurs de type interrupteur/potentiomètre permettant à l'utilisateur d'émettre une commande (commande essuyage pare-brise) ou d'acquérir une mesure (jauge à carburant ou la position du papillon des gaz),
- de simples contacteurs pour détecter des fins de courses (fin d'essuyage du pare-brise),
- des capteurs de pression, et de température (température d'eau, d'air d'admission...),
- ou encore des capteurs à effets Hall (capteurs de vitesse),
- ...

2.1.2.2 Les actionneurs

Les actionneurs les plus répandus sont les électrovannes et les moteurs électriques. Les signaux de commande provenant des entrées-sorties des calculateurs varient suivant le type d'actionneurs. Il peut s'agir de signal électrique continu ou d'un signal tout ou rien...

2.1.3 Architecture générale d'un réseau

La multiplication des fonctionnalités offertes à bord conduisant à des faisceaux électriques d'une complexité extrême, l'introduction du bus multiplexé a été vue comme un moyen de simplifier l'architecture électrique et donc d'en augmenter la fiabilité. La figure 2.2) représente le câblage qu'il faudrait avoir si le multiplexage n'existait pas. Chaque capteur devrait être connecté aux calculateurs qui ont besoin de l'information fournie par ce capteur. Par exemple, la figure 2.3 présente l'évolution du câblage en terme de nombre de points de connexion et de longueur de câble dans un véhicule. Le bus de communication permet d'échanger l'ensemble des informations dont les calculateurs ont besoin par l'intermédiaire d'un seul câble.

Un bus multiplexé est un réseau de communication qui permet aux différents calculateurs de s'échanger des informations au travers des messages circulant sur celui-ci. Chaque information est véhiculée sur le bus à l'aide d'un message de format défini mais de longueur variable (et limitée). Dès que le bus est libre, n'importe quel nœud relié au réseau peut émettre un nouveau message (cf. figure 2.4). Ainsi, les calculateurs en s'échangeant des messages par l'intermédiaire du réseau de communication peuvent remplir les fonctionnalités demandées.

De ce fait, un véhicule automobile actuel se voit doté d'un équipement informatique distribué qui, par sa structure, se rapproche de la complexité d'un réseau local informatique

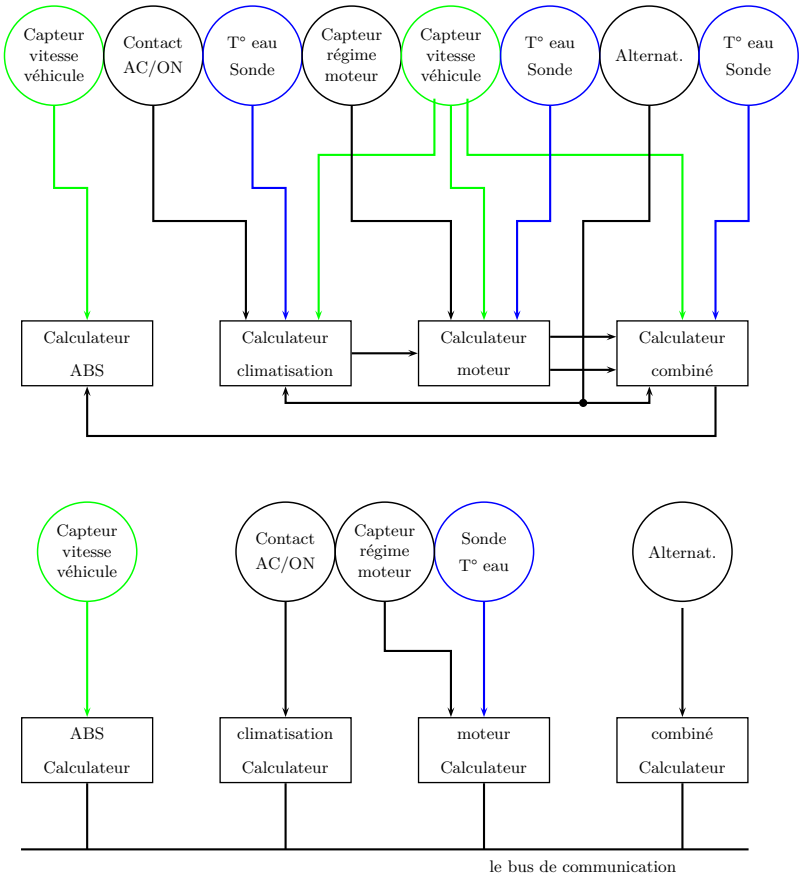


Figure 2.2 — Le multiplexage automobile

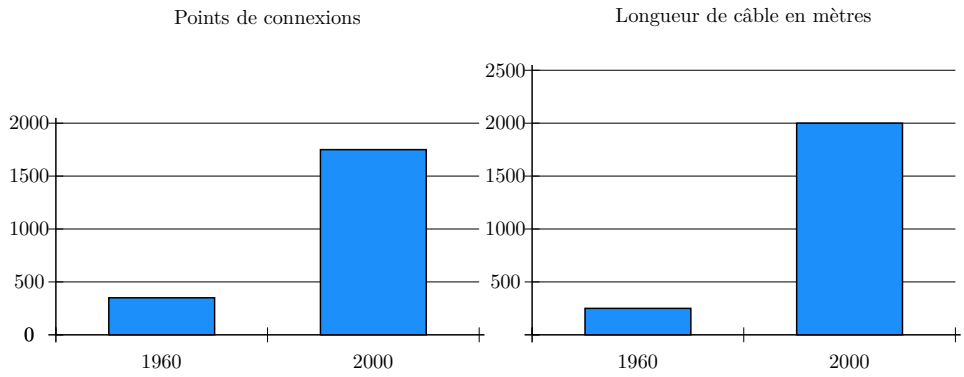


Figure 2.3 — Évolution du câblage

équipant le système de gestion d'une PME : certains véhicules actuels comportent une centaine de calculateurs embarqués reliés par un bus CAN (Controller Area Network). Bien sûr, les fonctionnalités offertes par cette informatique embarquée n'ont pas grand-chose à voir avec celles d'un réseau domestique, mais les aspects temps réel et sûreté de fonctionnement y sont particulièrement critiques, ce qui en augmente encore la complexité.

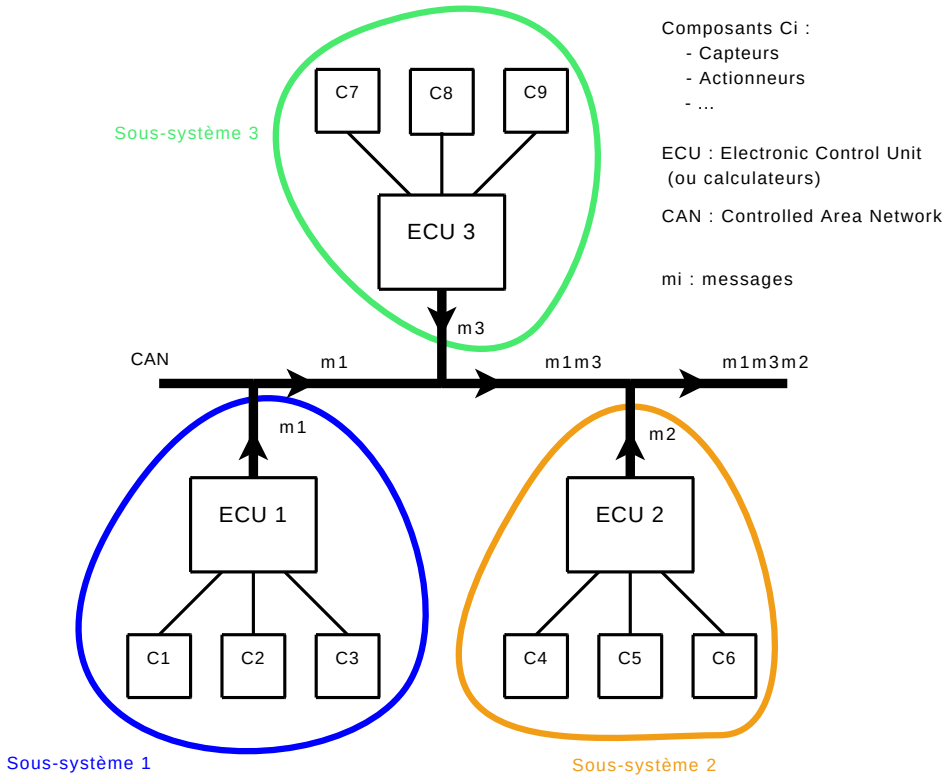


Figure 2.4 — Exemple de multiplexage

Chez le constructeur, cette complexité grandissante remet en cause les processus de développement et, alors que les problèmes de maintenance et de service après vente n'étaient pas considérés comme des services prioritaires, il apparaît aujourd'hui clairement qu'ils doivent être pris en compte très en amont, et être complètement intégrés dans la chaîne de développement.

2.2 Le diagnostic dans le domaine automobile

Les métiers de la maintenance automobile sont également affectés en profondeur par cette mutation (figure 2.5). Dans les années soixante, le dépanneur pouvait intervenir sur un véhicule en panne en s'armant d'une simple caisse à outils. Actuellement, un équipement spécifique capable de se connecter au système informatique embarqué pour en extraire les informations disponibles est nécessaire pour envisager un diagnostic de l'état du véhicule. Les méthodes de diagnostic employées dans ces outils sont elles-mêmes à définir pour assurer couverture et précision maximales du diagnostic tout en maîtrisant les coûts et délais de conception et de maintenance de ces outils de diagnostic.

2.2.1 Les fautes/pannes dans le domaine automobile

L'électronique rend la voiture plus fiable dans la mesure où les systèmes de sécurité sont plus précis et plus performants mais rend difficile la recherche de défaillances. Prenons

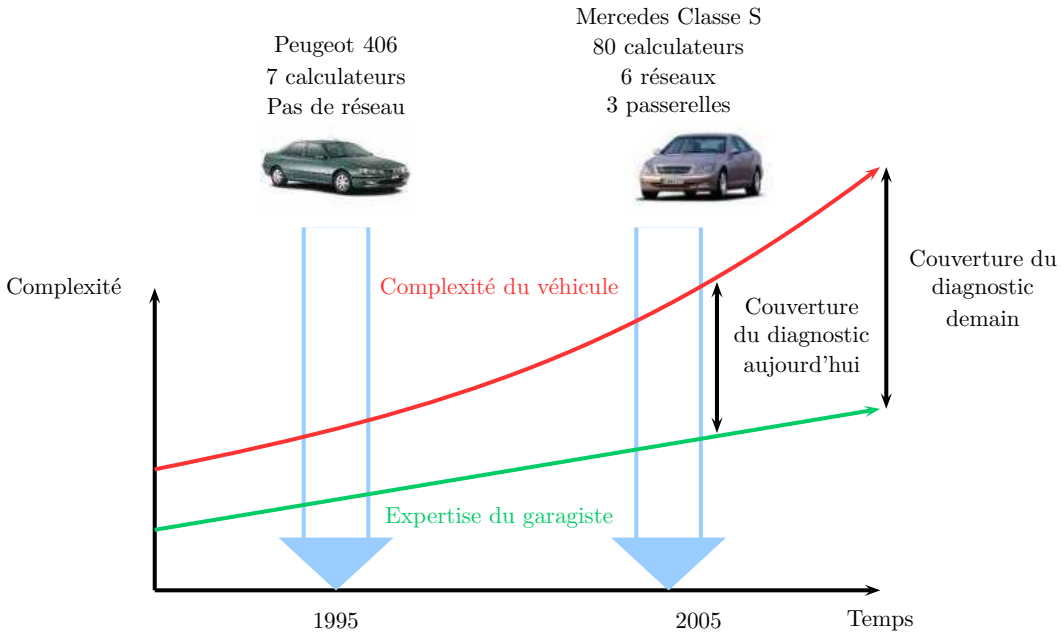


Figure 2.5 — Accroissement de la complexité du véhicule

comme exemple le cas d'une panne retrouvée chez un garagiste. Lorsque le moteur tournait et que la climatisation était enclenchée, l'appuie sur le bouton de commande du pulseur d'air provoquait un mouvement lent des essuie-vitres avant et arrière. Il s'agissait en fait d'un mauvais contact au niveau de la masse. Trouver la cause d'une défaillance devient très difficile pour le garagiste.

En outre, une difficulté majeure rencontrée aujourd'hui est l'apparition des défaillances fugitives et intermittentes. Ce sont des défaillances qui apparaissent puis disparaissent (et par conséquent leurs symptômes également). Nous prendrons l'hypothèse, souvent vérifiée en réalité, que ces défaillances ne sont donc plus présentes lors du diagnostic en garage. Ce type de défaillances est très difficile à diagnostiquer pour les garagistes dans la mesure où leurs moyens de diagnostic sont limités et ne sont plus adaptés aux défaillances actuelles.

Il existe plusieurs types de pannes :

- pannes électriques : elles sont essentiellement dues à des courts-circuits, des circuits ouverts ou des résistances parasites...
- pannes électroniques : fortement liées aux pannes électriques, ce sont toutes les pannes concernant les dispositifs électroniques (calculateurs, capteurs électroniques ...),
- pannes mécaniques : ce peut être des ruptures d'éléments, des frottements excessifs...
- pannes hydrauliques : il s'agit surtout des fuites...

Actuellement, ces types de pannes sont étudiés sur des sous-systèmes matériels spécifiques mais les systèmes électroniques dans le véhicule se multipliant, cela devient difficile de trouver des approches permettant une détection et/ou un diagnostic spécifique à chaque sous-système du véhicule. En outre, l'évolution de l'électronique embarquée s'accompagne de l'apparition de nouvelles défaillances qui sont, par conséquent, inconnues du garagiste ou des constructeurs. Or, la plupart des approches de diagnostic se base sur un modèle physique du système,

ce qui les rend difficilement applicables. Par contre, les modèles fonctionnels, généralement distribués sur les différents calculateurs, sont toujours connus et sont souvent réutilisés d'un type de voiture à l'autre, ou sont semblables entre les différentes marques de véhicules.

2.2.2 Le diagnostic en garage

Pour le diagnostic des véhicules, différents types de connaissances sont utilisés par les garagistes. Ces connaissances peuvent être classées de deux façons : les connaissances propres aux véhicules et les différentes informations provenant du véhicule lui-même.

2.2.2.1 Les différentes sources de connaissances

Différentes sources de connaissances sont disponibles pour enrichir une méthode de diagnostic. Ces sources de connaissances sont les suivantes :

Base d'incidents. Le garagiste a accès à une base des fiches des incidents connus. Chaque fiche incident mentionne les conditions d'applicabilité (modèle, type, motorisation, date de fabrication, options, etc.), des symptômes clients observables sur le véhicule, un diagnostic et un mode opératoire permettant de réparer le véhicule.

Modèles de bon fonctionnement. Le garagiste utilise des documents papier (ou informatique) lui donnant des connaissances sur le bon fonctionnement du système. Par exemple, les revues techniques mentionnent une valeur (souvent une mesure de résistance) pour tester si un composant est dans son état normal. Les principes de fonctionnement (fichier PDF) décrivent le comportement nominal d'un système.

Ensemble de fonctions. L'ensemble des fonctions est décrit par une décomposition de l'architecture du véhicule. Par exemple, nous distinguons les fonctions "Essuyage/Lavage", "Climatisation", "Signalisation"...

Réseau causal. Le réseau causal est un réseau sémantique permettant de relier les symptômes aux fonctions dont la défaillance peut être la cause de leur apparition.

2.2.2.2 Les différentes sources d'informations provenant du véhicule

En plus de ces connaissances provenant de sources extérieures, il existe d'autres sources d'informations en provenance du véhicule pour accomplir la tâche de diagnostic. Nous retrouvons comme sources d'informations :

Description des symptômes/effets client. La première source d'information en vue d'établir un diagnostic est la description par le client des effets perceptibles de la faute sur le véhicule. Cette description est notée de manière textuelle dans une fiche remplie par le réceptionniste. Cette fiche est ensuite transmise au garagiste pour qu'il effectue son diagnostic.

Contexte du véhicule. Le garagiste utilise le code VIN (Vehicle Identification Number) du véhicule et à travers un accès à la base de données "véhicule" du constructeur il récupère toutes les informations contextuelles sur ce véhicule (type, motorisation, options, date de fabrication, etc.)

Codes défauts présents sur les calculateurs. Le garagiste utilise les codes défauts qui sont présents sur les calculateurs. Ces codes sont lus avec un outil de diagnostic.

Données issues d'un test de roulage. Si la description des symptômes présents sur le véhicule n'est pas claire ou si le garagiste veut tester un point complémentaire, il effectue un test de roulage avec le véhicule de manière à affiner la description des symptômes perceptibles. De surcroît, si nous voulons obtenir plus d'information concernant des défaillances intermittentes, il est nécessaire d'effectuer ce test de roulage. C'est pourquoi, embarquer un module de diagnostic pour faire des tests sur le véhicule est un moyen qui est de plus en plus envisagé. Si ce module était embarqué de façon permanente dans le véhicule, il fournirait de nombreuses informations utiles au garagiste pour son diagnostic. C'est ce qu'envisage de faire le projet DDP (Diagnostic Distribué et Préventif) que nous aborderons un peu plus loin (paragraphe 2.3.2.4).

Résultat Mesure. Le garagiste peut effectuer la mesure d'une grandeur physique sur le véhicule (tension, résistance, pression, etc.)

Réparation. Ce sont les procédures à appliquer, associées aux fautes identifiées, pour réparer le véhicule.

Les moyens pour établir un diagnostic ont évolué ces dernières années mais ils ne permettent pas encore de résoudre tous les problèmes rencontrés dans une voiture. Notamment, pour les défaillances fugitives et intermittentes, peu de connaissances peuvent être utilisées :

- il n'existe pas ou peu de base d'incidents,
- les mesures ne permettent pas d'établir un diagnostic puisque par définition la défaillance n'est plus présente,
- le mode roulage est ponctuel. Il faut que la défaillance apparaisse durant cette phase,
- ...

En outre, toutes ces données sont utilisées de façon désordonnée sans réelle stratégie de diagnostic. Il convient donc de mettre en œuvre une méthode qui puisse prendre en compte un maximum d'information pour arriver à un diagnostic précis du système. Le diagnostic dans le domaine automobile pose ainsi une vraie problématique scientifique à laquelle se dédie le laboratoire commun AutoDiag qui réunit deux laboratoires toulousains, le LAAS-CNRS et l'IRIT, aux cotés de la société ACTIA (site web : http ://www.laas.fr/autodiag/).

2.3 Le projet MODE

2.3.1 Structure générale

Le cadre général du projet mené par le LAAS-CNRS conjointement avec l'IRIT et la Société ACTIA, porte sur le diagnostic dans le domaine appelé aujourd'hui la "mécatronique". Son objectif est de réaliser une étude et le prototype d'un système de diagnostic utilisant tous les types de données accessibles. Ce système de diagnostic est nommé "MODE" (figure 2.3.1).

Face à la complexité croissante des véhicules et à la diversité des systèmes embarqués, il apparaît nécessaire de concevoir un système de diagnostic faisant collaborer plusieurs méthodes afin d'utiliser le maximum de connaissances disponibles sur le système et de maximiser le taux de couverture en terme de types de fautes pouvant apparaître sur ce système. Trois thèses ont été mises en place pour développer ces différentes méthodes.

Par ailleurs, lors de la conception d'un système de diagnostic industriel, un facteur clé de viabilité est le coût, qu'il soit de développement, de maintenance ou de génération des données de diagnostic nécessaires. Ainsi, afin de maximiser la couverture du diagnostic tout en minimisant les coûts, le système de diagnostic MODE utilise plusieurs types d'informations traitées par les méthodes adaptées et met en œuvre une stratégie de coopération de telle sorte que le rapport pertinence des données utilisées dans une méthode de diagnostic sur coût d'obtention des informations soit maximal.

Un tel système présente sans conteste de nombreux aspects innovants. D'une part, il s'agit de trouver, par une représentation pertinente des données, un équilibre dans la prise en compte et le traitement de tous les types et niveaux d'information disponibles sur le système à diagnostiquer. D'autre part, il s'agit de développer des techniques de diagnostic puissantes permettant de caractériser et de diagnostiquer des systèmes complexes, de manière robuste vis-à-vis de leur environnement et rendant possible un diagnostic en conditions d'utilisation habituelles. Enfin, il s'agit de développer une démarche méthodologique visant à la mise en place des algorithmes de diagnostic dans un contexte opérationnel prenant explicitement en compte la présence de l'utilisateur.

2.3.2 Les différents projets de recherche de MODE

2.3.2.1 Le projet OBIR : Ontology Based Information Retrieval

Afin d'aider un garagiste dans sa tâche de diagnostic, la thèse entreprise a pour ambition de concevoir un module de gestion de base d'expériences.

L'application consiste à rechercher dans une base d'expérience des fiches de réparation pertinentes pour une panne sur un véhicule donné : à partir de la description en langue naturelle des symptômes, l'outil doit retrouver un ensemble de fiches structurées traitant d'un problème similaire sur le même type de véhicule. Il a été choisi d'adapter des technologies du web sémantique dans ce cadre (domaine spécialisé, documents très structurés et utilisateur ayant une tâche de diagnostic à réaliser), à savoir de s'appuyer sur un modèle de connaissances (une ontologie) pour représenter le contenu des documents et y rechercher des informations.

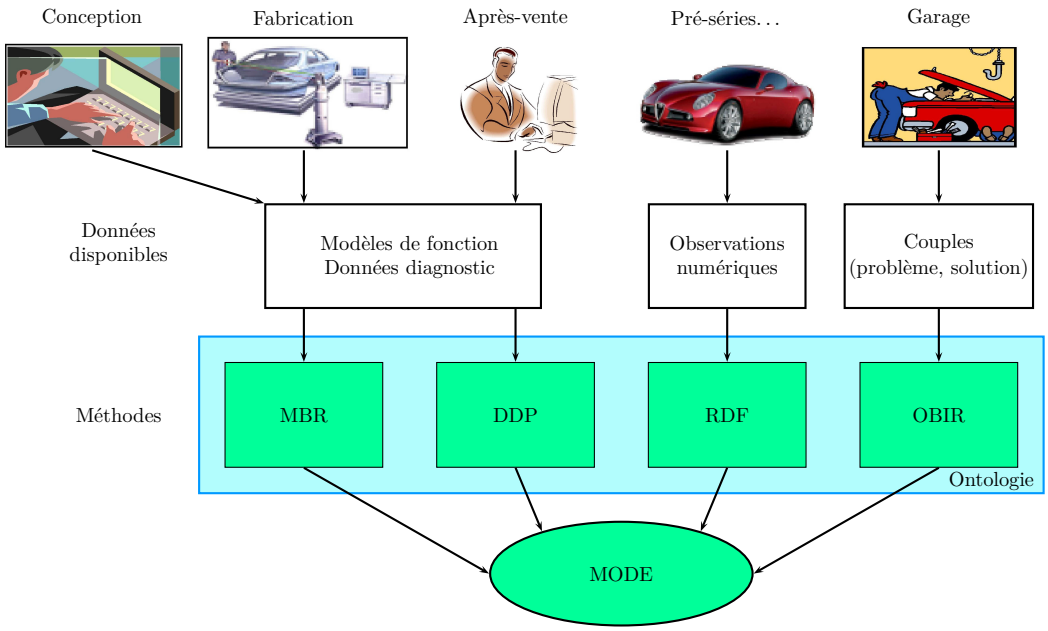


Figure 2.6 — L'approche MODE

Un simple moteur de recherche basé sur une indexation classique par mots-clés aurait pu suffire mais, en l'absence d'une modélisation des connaissances du diagnostic automobile, il aurait été impossible de raisonner sur les objets du domaine (e.g. la présence d'un symptôme peut entraîner celle d'un autre).

Les efforts ont été concentrés sur la mise au point d'un mécanisme d'indexation conceptuelle des fiches et d'une chaîne de traitement (notamment avec des outils de Traitement Automatique du Langage) visant à maintenir le modèle de connaissances lors de l'ajout de nouvelles fiches dans la base d'expérience.

Ce module exploite les fiches incidents disponibles pour chaque type de véhicule, et par conséquent, les symptômes/effets clients, le contexte du véhicule, le réseau causal, et la réparation.

L'ensemble de ces travaux fait l'objet de publications (Reymonet *et al.*, 2006; Reymonet *et al.*, 2007*a*; Reymonet *et al.*, 2007*b*).

2.3.2.2 Le projet MBR : Model Based Reasoning

Dans le domaine du diagnostic hors ligne, comme celui effectué dans les garages dans le domaine de l'automobile, un problème essentiel est celui de la détermination de la séquence de tests et mesures aux points de contrôle, qui permettent de localiser la faute le plus rapidement possible et avec un coût minimal. Ce problème est connu sous le nom de Test Sequencing Problem.

La thèse de (Faure, 2001) (sous convention CIFRE avec ACTIA) a abouti à une méthode opérationnelle, appelée AGENDA (Automatic GENeration of DiAgnosis trees) de résolution du Test Sequencing Problem. La solution proposée génère la table des signatures des fautes à

partir de modèles de fautes ensemblistes qui prennent en compte les tolérances de conception des composants. Un algorithme de recherche de type AO* associé à une heuristique permet de générer l'arbre de diagnostic optimal. Cette optimalité tient compte de coûts dynamiques pour les tests. En effet, le coût d'un test doit être mis à jour au fur et à mesure des déposes qui sont effectuées.

La suite du travail (Olive, 2003) (réalisé dans le cadre d'une autre convention CIFRE avec ACTIA) a consisté à étendre l'applicabilité d'AGENDA et à améliorer les performances de cette approche. Les concepts de modélisation d'AGENDA ont été étendus à des composants à plusieurs modes de fonctionnement.

Aujourd'hui, le module MBR est un module qui doit prendre en compte plusieurs domaines techniques et proposer différentes approches de diagnostic (hors ligne, interactif, en ligne).

Les axes développés sont les suivants :
- Élargissement du champ d'action de l'approche MBR par la prise en compte d'autres domaines que le domaine électrique (domaines mécaniques, thermiques, écoulement d'air, hydrauliques...).
- Développement d'une stratégie de haut niveau (type fonctionnel) afin de réaliser du diagnostic (étape de localisation) au niveau d'une fonction complète.

Le travail porte donc sur l'élaboration d'une stratégie de raisonnement multi-modèles pour faire collaborer divers types de modèles, exploiter différents points de vue et gérer les redondances éventuelles. En effet, selon les différentes parties du véhicule considérées, différents modèles sont disponibles (modèles mécaniques, modèles physiques...) de même que la connaissance experte et le savoir-faire des ingénieurs de l'automobile.

Ce module prend en compte les symptômes/effets client ainsi que les codes défauts des calculateurs, les ensembles de fonctions ou encore les résultats mesures et les modèles de bon fonctionnement. Il se focalise essentiellement sur des défaillances persistantes. L'ensemble de ces travaux fait l'objet de publications (Ressencourt *et al.*, 2006; Resencourt, 2006).

2.3.2.3 Le projet RDF : Reconnaissance Des Formes

L'objectif de ce projet est de mettre au point un outil de diagnostic de pannes utilisant des techniques de reconnaissance des formes (cf. chapitre 1). Il vise donc à améliorer les capacités d'autodiagnostic des calculateurs en prenant en compte les corrélations entre paramètres.

Les objectifs de ce module sont triples :
- Pallier certaines limites des fonctions d'auto-diagnostic des calculateurs. En effet, même si l'auto-diagnostic des calculateurs est de plus en plus évolué, de trop nombreuses pannes ne sont pas encore détectées ou le code défaut remonté par le calculateur est trop général pour permettre au garagiste de trouver l'origine de la panne.
- Diagnostiquer aussi bien des pannes d'origine électronique que mécanique. Un des principaux intérêts de ce module est de permettre l'isolation de panne d'origine mécanique dès que la panne influe sur une grandeur physique mesurée par un capteur.
- Réduire l'effort de modélisation et d'expertise, car contrairement à d'autres méthodes de diagnostic, la RDF repose sur l'utilisation de techniques d'apprentissage automatique pour créer les modèles du véhicule. Ces modèles sont de ce fait beaucoup moins coûteux

à élaborer que des modèles basés sur l'avis d'experts.

Le principe de fonctionnement repose sur la corrélation de paramètres. Ce projet consiste à surveiller des paramètres prédéfinis et à détecter (voire localiser si connues) des défaillances connues (ou inconnues si le système sort simplement du bon fonctionnement). Il se focalise sur un sous-système spécifique comme le moteur dont le modèle mathématique est difficile à obtenir. Celui-ci sera donc créé par apprentissage lors de test de roulage.

2.3.2.4 Le projet DDP : Diagnostic Distribué Préventif

Un module de diagnostic préventif peut s'envisager en tant que système mis à disposition chez le garagiste et utilisé systématiquement lors du passage d'un véhicule au garage. Mais le diagnostic préventif prend cependant toute sa portée lorsqu'il s'inscrit dans le triangle véhicule, constructeur, garage et qu'il est envisagé au travers d'un système embarqué. A bord et en temps réel, la notion de gravité d'une situation anormale prend alors une importance particulière ; il faut être en effet capable d'informer le conducteur de la nécessité se rendre rapidement au garage, voire d'actionner automatiquement un coupe-circuit pour isoler un organe non vital défaillant. La présence à bord du système de diagnostic permet aussi de détecter et d'isoler lorsqu'elles sont présentes les défaillances fugitives ou intermittentes et d'enregistrer le contexte d'apparition des défaillances.

La surveillance de paramètres continus pour faire de la détection n'est pas le sujet de ce module. Cet aspect est très largement couvert dans la littérature relative au diagnostic (cf. chapitre 1) et est abordé dans AUTODIAG par le module de diagnostic par reconnaissances des formes (paragraphe 2.3.2.3).

Le cas des défaillances persistantes fait déjà l'objet de travaux complexes dans le projet MBR (paragraphe 2.3.2.2). Pour des défaillances intermittentes, et donc lorsqu'elle ne sont plus présentes lors du diagnostic en garage, il est très difficile voire impossible pour le garagiste de localiser la défaillance ou d'en déterminer la cause. Les seules informations qu'il possède, sont les symptômes/effets client, les codes défauts des calculateurs quand il y en a, les modèles, et les fiches d'incident.

Il faut savoir que les codes défauts des calculateurs ne sont générés que par le calculateur et ne concernent que l'état de ses bornes ou des dépassements de seuil et violation de contraintes lors la surveillance de paramètres. Ces paramètres sont peu nombreux et ne concernent que des fonctions critiques. Tout le problème est de détecter une défaillance car un code défaut est associé à un seul capteur ou actionneur.

Les fiches d'incident peuvent être utiles si la défaillance est déjà apparue et qu'un diagnostic a été établi.

Les mesures qui sont finalement le moyen de diagnostic le plus utilisé dans le projet MBR sont souvent inutiles pour des défaillances intermittentes.

Ce type de défaillances est donc un réel problème pour l'industrie automobile. Pour pouvoir les détecter et ensuite les diagnostiquer, il est nécessaire d'établir une approche pour un diagnostic embarqué. Pour cela, nous avons plusieurs possibilités. Nous pouvons comme pour le projet RDF nous focaliser sur un sous-système et surveiller des paramètres lorsque le modèle du sous-système est inconnu. Si nous avons un modèle de celui-ci, nous pouvons

l'utiliser et faire du diagnostic à base de modèle comme nous l'avons vu dans le chapitre 1. Néanmoins, l'utilisation de telles techniques nécessite de savoir quels sont les sous-systèmes à surveiller. Or avec la multiplication des sous-systèmes électroniques et leurs interactions, il est difficile de connaître les sous-systèmes qui peuvent influer les uns sur les autres. Il est également difficile de connaître à l'avance la localisation d'une défaillance et de déterminer les paramètres à surveiller.

Ce module est par conséquent dédié à la surveillance du comportement des fonctions véhicules (essuyage, lavage etc...) au travers des informations accessibles en temps réel sur tout type de véhicule. L'architecture "multiplexée" actuelle des véhicules comporte un ou plusieurs réseaux CAN. L'interaction des différents calculateurs de commande embarqués utilise ce réseau. Dans notre approche, le réseau CAN est donc le point d'observation privilégié de cette interaction.

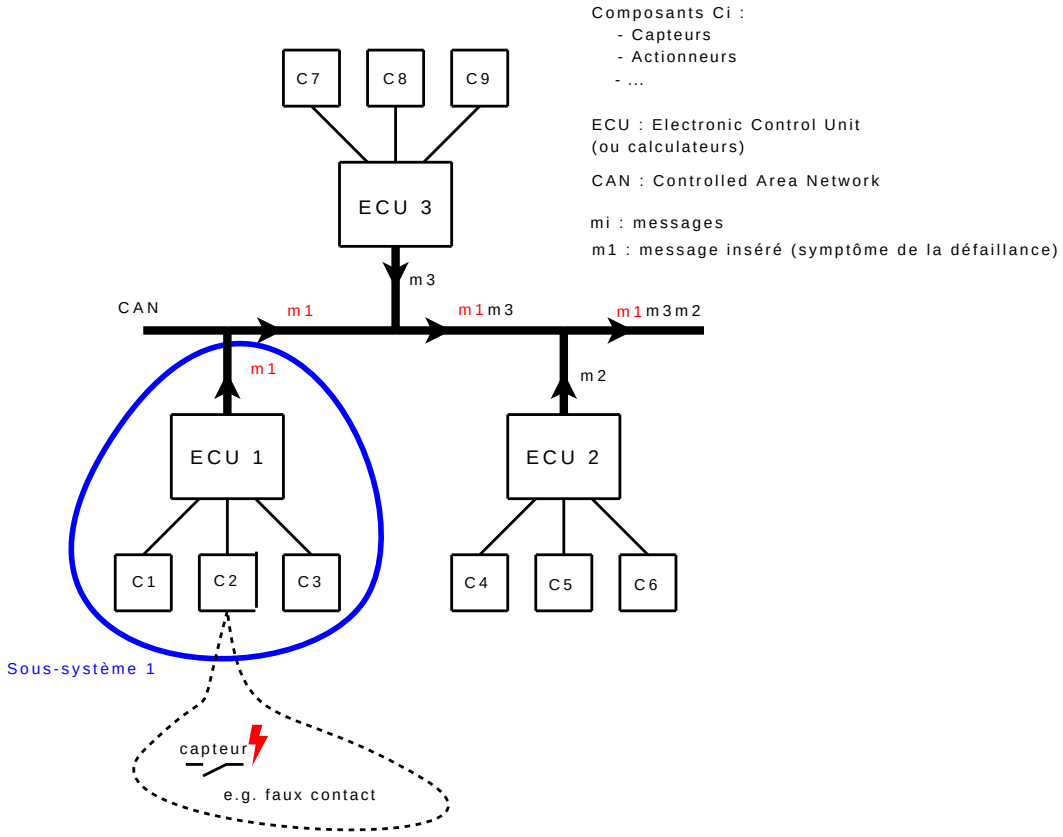


Figure 2.7 — Exemple de symptôme d'une défaillance fugitive sur le réseau de communication

Certains défauts fugitifs et intermittents sont caractérisés par l'absence ou l'insertion inopinée d'un message sur le réseau (cf. figure 2.7). Nous ne cherchons donc pas à détecter l'ensemble des défauts fugitifs et intermittents mais seulement ceux qui se traduisent par les symptômes énoncés ci-dessus. Cette classe de défaut englobe les faux contacts, les parasites, etc. L'émission inopinée d'un message par un calculateur (ou boîtier électronique) ou l'absence d'un message peut provoquer un dysfonctionnement du système électronique embarqué. Le comportement erratique qui en résulte souvent n'est généralement plus observable lorsque le véhicule est amené au garage par le client. Ce type de défaut ne peut être détecté que par

le module de diagnostic embarqué qui peut effectuer simultanément une pré-localisation de la défaillance en désignant le calculateur mis en cause ainsi que les différents composants qui lui sont connectés. Cette pré-localisation permettra par la suite, dans le garage et hors ligne, de faire un diagnostic plus détaillé afin de déterminer la cause du problème.

2.3.3 Conclusion

Nos travaux consistent donc à détecter, localiser un message qui serait issu d'une défaillance fugitive ou intermittente. Nous nous plaçons donc dans une problématique de systèmes à événements discrets puisque nous nous intéressons à l'échange de données entre les différents calculateurs. En outre, nos travaux se basent sur les données disponibles à savoir les modèles de bon comportement des fonctions puisque c'est la seule connaissance que nous avons (pas de modèle des défaillances intermittentes). Nous verrons dans le chapitre suivant les différentes approches qui ont été utilisées pour le diagnostic des systèmes à événements discrets ainsi que celles se focalisant sur les défaillances intermittentes.

3 Diagnostic à base de modèle pour les systèmes à événements discrets

Introduction

Comme nous l'avons vu dans le chapitre 2, notre idée est d'utiliser une modélisation à événements discrets pour le diagnostic des fautes fugitives et intermittentes dans le domaine automobile. Ce genre d'approche a été très peu utilisé dans les systèmes embarqués comme nous avons pu le constater dans le chapitre 1.

Ce chapitre a pour objectif de donner une vision générale des méthodes de diagnostic appliquées aux systèmes à événements discrets, sans prétendre être exhaustif. Nous distinguons deux cas : les approches à base de modèles de bon comportement et les approches à base de modèles de fautes. Même si les approches à base de modèles appliquées aux systèmes à événements discrets utilisent ces deux types de modèles, les approches à base de modèles de fautes sont les plus couramment étudiées, avec notamment l'approche la plus connue : le diagnostiqueur. Nous présentons ensuite les travaux qui ont été menés sur le diagnostic des défaillances fugitives ou intermittentes. Une synthèse est ensuite faite dans laquelle nous positionnons nos travaux pour un diagnostic embarqué des défaillances intermittentes à base de modèles à événements discrets.

3.1 Approches de diagnostic à base de modèles de bon comportement

Les approches à base de modèles de bon comportement restent très peu employées pour les systèmes à événements discrets. Le principe de ces approches est de comparer les événements reçus (les observations) du système surveillé avec ceux attendus par le modèle décrivant le bon comportement du système. La détection survient lorsqu'il existe une incohérence entre ces deux types d'événements (attendus et reçus). Ainsi, toute séquence qui n'est pas générée par le modèle est considérée comme une déviation anormale du comportement. Dans l'ensemble des approches, il est généralement considéré que le symptôme détecté c'est-à-dire l'événement qui est incohérent avec le modèle résulte d'une défaillance affectant un composant du système.

L'information sur les symptômes est importante pour le diagnostic car elle permet de déterminer les éléments susceptibles d'être défaillants pouvant expliquer l'incohérence constatée entre les observations et le modèle.

L'utilisation de modèles de bon comportement permet de déterminer les comportements attendus et corrects du système. Les modèles utilisés dans ce cadre là sont souvent des modèles exploitant les informations temporelles du système. La plupart des modèles de systèmes à événements discrets représentent le comportement du système évoluant d'un état à un autre en fonction d'événements reçus. L'occurrence de ces événements est alors associée à une fenêtre temporelle. Si l'événement est reçu en dehors de cette fenêtre, alors il est considéré comme la manifestation de la présence d'une défaillance. C'est le principe de détection, appliqué dans la majorité des approches utilisant des modèles de bon comportement (Holloway et Krogh, 1990; Combacau, 1991; Valette et Kunzle, 1994).

L'approche des chroniques se base également sur le même principe de détection mais modélise non plus les états du système mais donne une description des événements contraints temporellement entre eux. Les chroniques sont généralement utilisées avec des modèles de fautes mais nous pouvons également les utiliser avec un modèle de bon comportement. Les chroniques sont constituées d'un ensemble d'observations et d'un ensemble de contraintes temporelles entre les instances d'occurrence de celle-ci. Cet ensemble de contraintes temporelles est représenté sous la forme d'un graphe d'instants (cf figure 3.1).

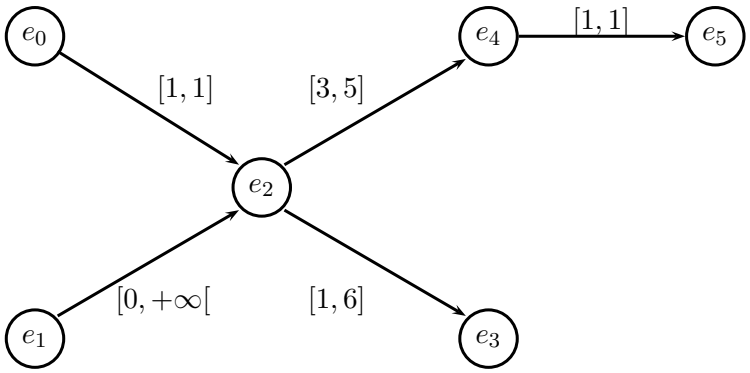


Figure 3.1 — Graphe des instants d'un modèle de chroniques

Dans l'exemple de la figure 3.1, chaque événement e_i est associé à une occurrence parmi les événements observés. La transition du graphe étiquetée $[t_{min}, t_{max}]$ entre un événement e_i et un événement e_j représente la contrainte temporelle "si e_i survient à la date t_i alors e_j survient à la date t_j telle que $t_i + t_{min} \leq t_j \leq t_i + t_{max}$". La surveillance d'un système à l'aide de chroniques est fondée sur la reconnaissance en ligne de modèles de chroniques. Le principe est le suivant : à chaque observation, un ensemble d'instances de modèle de chroniques est maintenu pour lesquelles l'ensemble des observations reçues à cet instant est compatible avec le graphe des instants de chaque chronique. À la réception d'une nouvelle observation les chroniques qui ne sont pas compatibles (typiquement l'observation est reçue trop tôt ou trop tard suivant ce scénario) sont éliminées de cet ensemble et les chroniques

qui peuvent débuter avec l'arrivée de cette nouvelle observation sont ajoutées. Une chronique est reconnue lorsque tous les événements représentés dans le graphe des instants ont eu lieu dans l'ordre et les délais notifiés dans le graphe. Des outils de reconnaissance de chroniques mettent en œuvre ce principe : IxTeT (Mounir-Alaoui, 1990; Dousson *et al.*, 1993), et son successeur CRS (Chronicle Recognition System) dont l'objectif principal est l'efficacité en ligne de la reconnaissance.

À partir de cette détection, différentes méthodes sont employées pour établir un diagnostic. La stratégie de diagnostic consiste à déterminer les violations de contraintes, le ou les composants défaillants qui ont émis l'événement, ou encore les conditions opératoires qui ne sont plus satisfaites, etc. qui peuvent expliquer le comportement anormal observé.

Lorsque les informations temporelles ne sont pas disponibles ou ne sont pas pertinentes, les techniques utilisées s'appuient sur un raisonnement basé sur la structure même du modèle.

Le principe de détection est basé sur la détection des violations des propriétés du modèle. Par exemple, dans les réseaux de Petri, il est impossible de franchir une transition qui n'est pas franchissable (Pradin-Chezalviel et Valette, 1993; Valette et Kunzle, 1994) ou encore les propriétés structurelles des invariants de places doivent être satisfaites (Tabakow, 2007; Hadjicostis et Verghese, 1999), ...

Le principe de diagnostic repose sur la recherche de toute séquence de franchissements de transitions capable de faire évoluer le réseau de Petri depuis le dernier marquage connu jusqu'à un marquage sensibilisant la transition à franchir en utilisant la logique linéaire introduite par (Girard, 1987) pour faire un raisonnement arrière (Pradin-Chezalviel et Valette, 1993; Valette et Kunzle, 1994). Le principe de diagnostic peut également consister à déterminer les places dont un marquage erroné expliquerait le marquage observé et la violation des propriétés (Tabakow, 2007) ou dans une plus large mesure à déterminer la production de jetons dans des places de sorties sans consommation des jetons des places d'entrée, la consommation des jetons des places d'entrée sans production des jetons en places de sortie ou encore la corruption du nombre de jeton d'une place (Hadjicostis et Verghese, 1999). Une bonne connaissance est cependant nécessaire pour expliquer les manifestations d'une défaillance du système sur le modèle et cette défaillance.

L'utilisation de modèles de bon comportement est peu adoptée pour le diagnostic des systèmes à événements discrets. Il est vrai que ce genre d'approches ne permet pas de faire un diagnostic précis du système dans la mesure où la localisation de la défaillance est difficile sans modèle de fautes. La détection se base sur des violations de contraintes temporelles ou alors des propriétés structurelles du modèle. Le diagnostic consiste à déterminer les conditions opératoires qui expliquent les incohérences observées ou alors les places, ou transitions, qui, par leurs marquages ou leurs franchissements, permettraient d'expliquer les violations structurelles observées. Les approches de diagnostic ne sont généralement pas adaptées pour le diagnostic en ligne mais l'utilisation de modèles de bon comportement permet de faire de la détection et du diagnostic de système lorsque la connaissance des défaillances n'est pas suffisante pour en établir un modèle.

3.2 Approches de diagnostic à base de modèles de fautes

Dans les systèmes à événements discrets, les approches les plus utilisées sont les approches à base de modèles de fautes. L'approche la plus connue est celle dite du "diagnostiqueur" introduite dans (Sampath *et al.*, Sep 1995; Sampath *et al.*, 1996). Un diagnostiqueur est un automate obtenu par compilation d'un modèle contenant des événements de fautes (inobservables) et dont les événements déclenchant les transitions sont les événements observables du système. L'état du diagnostiqueur donne par conséquent, à tout instant, une information sur les fautes pouvant expliquer le comportement observé. Le diagnostic consiste à identifier ces événements inobservables (les fautes). La détection et l'identification d'une faute se fait sur le même automate. En effet, la détection d'une faute se fera lorsqu'il n'existera plus d'ambiguïté entre l'hypothèse du comportement normal et celle d'un comportement de faute.

Sur la figure 3.2, un exemple de diagnostiqueur (à droite) est représenté. Il est établi à partir d'un automate global (automate de gauche) dont l'état initial est 1. Sur cet automate global sont représentés des événements observables σ_i, des événements non observables σ_{uo}, et un événement de faute (non observable) σ_{f1}. Pour le diagnostiqueur, chaque état est constitué de plusieurs états de l'automate global correspondant aux différents modes de défaillances. Nous associons à ces états des labels : N pour décrire le comportement normal, F_i pour décrire qu'une défaillance F_i a eu lieu. À l'état initial, le diagnostiqueur est dans l'état 1 et aucune faute ne s'est produite (label N). Si σ_1 est observé, le modèle est soit dans l'état 7, correspondant à un comportement normal (7N), soit dans l'état 3, correspondant à l'occurrence d'une défaillance de type f_1 ($3F_1$). Il existe donc une ambiguïté entre le comportement normal et la possibilité de l'occurrence d'une défaillance (7N,$3F_1$).

Le calcul du diagnostiqueur se fait hors-ligne et nécessite la construction du modèle global. Le processus de diagnostic est quant à lui effectué en ligne et consiste à parcourir le diagnostiqueur en fonction des observations reçues et à donner l'information résumée dans l'état courant du diagnostiqueur.

Initialement dans (Sampath *et al.*, 1996), le diagnostiqueur était une approche centralisée chargée de surveiller, détecter et diagnostiquer les défaillances du système global. D'autres approches du diagnostiqueur centralisé peuvent être consultées dans (Ushio *et al.*, 1998; Chung *et al.*, 2003). Ces approches utilisent le formalisme des réseaux de Petri. (Ghazel *et al.*, 2005) proposent d'utiliser les informations temporelles pour réduire les résultats.

Or, avec des systèmes complexes, l'utilisation de modèles globaux s'est avérée vite problématique. C'est pourquoi beaucoup d'approches ont donc été menées pour résoudre le problème de la complexité des modèles. (Rozé et Laborie, 1998; Rozé et Cordier, 2002) proposent de régler ce problème en introduisant le concept de diagnostiqueur générique.

Nous retrouvons également des approches décentralisées basées sur le principe du diagnostiqueur. Par exemple, dans (Debouk *et al.*, 1998), leur approche est basée sur la construction de diagnostiqueurs locaux qui vont ensuite communiquer un diagnostic local à un coordinateur en charge d'établir un diagnostic global du système. Cependant, dans cette approche, la construction des diagnostiqueurs locaux se fait à partir d'un modèle global, ce qui est problématique pour des systèmes de grande taille.

Dans (Pencolé, 2002), les diagnostiqueurs locaux sont construits à partir de modèles

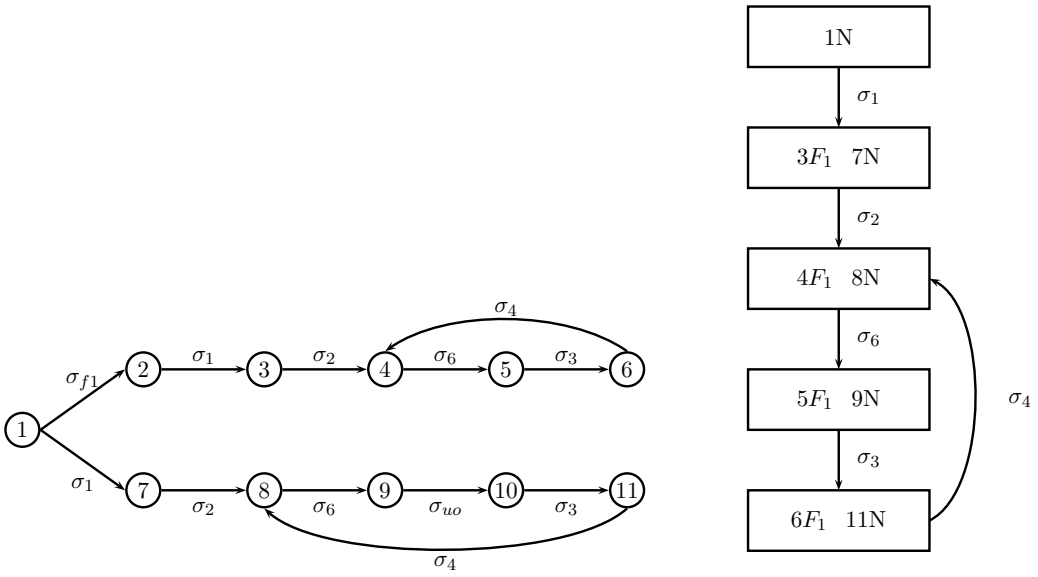


Figure 3.2 — Exemple d'un diagnostiqueur

locaux. Les informations de diagnostic sont ensuite communiquées à un coordinateur qui fusionne les diagnostics locaux pour établir un diagnostic global. Cette approche présuppose la connaissance a priori des observations. Une extension de ce travail est proposée dans (Grastien, 2005) pour faire du diagnostic en ligne. Le principe est de faire du diagnostic incrémental c'est-à-dire d'établir un diagnostic à partir des observations déjà reçues et de l'adapter en fonctions des nouvelles observations qui apparaissent.

(Mouchaweh *et al.*, 2006; Philippot, 2006; Philippot *et al.*, 2007) proposent une approche décentralisée qui utilise toutes les sources d'informations disponibles dans le but d'avoir un modèle détaillé du système et d'en réaliser le diagnostic. Les différentes sources exploitées sont :

- les informations opérationnelles caractérisant le comportement désiré (modèle de contrôle),
- les informations structurelles provenant du procédé et définissant l'emplacement des capteurs et des actionneurs (modèle structurel),
- les informations temporelles sur la réactivité des actionneurs, i.e. une fenêtre temporelle définissant les temps de réponses des actionneurs (modèle temporel).

Des diagnostiqueurs locaux (au sens de (Sampath *et al.*, 1998)) sont basés sur la combinaison de ces trois modèles dans le but de prendre une décision sur l'occurrence de fautes. Un coordinateur récupère toutes les décisions qu'il va combiner au moyen d'un opérateur booléen afin d'obtenir une décision globale.

D'autres approches dites distribuées sont également étudiées. Dans (Su *et al.*, 2002), les diagnostiqueurs locaux, établis à partir de modèles locaux, communiquent entre eux pour partager leurs informations de diagnostic avec l'objectif d'améliorer leurs diagnostics locaux. Le diagnostic global est déterminé par la recherche de la cohérence entre les diagnostics locaux.

(Genc et Lafortune, 2003; Genc et Lafortune, 2005) emploient un formalisme différent (les réseaux de Petri) mais se basent toujours sur le même principe que celui proposé par Sampath, à savoir l'ambiguïté d'états. Les transitions sont labellisées par des événements observables ou non, et de fautes (non observables). Des diagnostiqueurs locaux sont construits à partir de modèles locaux avec des places communes et peuvent communiquer entre eux après réception d'un événement lors de la phase en ligne pour mettre à jour leur état. L'ambiguïté d'état est représentée par différents jetons.

Dans (Jiroveanu et Boel, 2005*a*; Jiroveanu et Boel, 2005*b*), nous avons la même modélisation mais au lieu d'avoir un diagnostiqueur au sens de Sampath comme dans (Genc et Lafortune, 2005), les auteurs utilisent des agents diagnostiqueurs locaux qui effectuent un algorithme arrière. Cet algorithme recherche les séquences minimales dans le réseau de Petri et le nombre minimum de jetons nécessaires dans les places d'entrée permettant d'expliquer l'occurrence d'un événement. Ces diagnostiqueurs locaux communiquent entre eux afin d'être consistants avec les diagnostics des autres diagnostiqueurs locaux.

(Lefebvre et Delherm, 2005) utilisent également des réseaux de Petri incluant des défaillances représentées par des transitions non observables. La détection et le diagnostic sont traités à l'aide de relations de causalité et de chemins dirigés (directed paths) afin de déterminer les places et les transitions à observer pour le diagnostic de fautes.

Une autre méthode présentée dans (Giua, 1997; Giua et Seatzu, 2002) consiste à estimer l'état du marquage courant à partir des observations (le marquage pouvant représenter un état de défaillance).

(Benveniste *et al.*, 2003) développe une approche originale fondée sur l'utilisation de sémantiques d'ordre partiel du modèle dont l'objectif est de reconstruire les scénarii possibles du modèle cohérents avec les observations. Il s'agit d'utiliser le modèle pour inférer les relations de causalité et de concurrence entre les événements observés. Les ambiguïtés dans les observations conduisent à un ensemble d'explications possibles, représenté par la notion de dépliage des réseaux de Petri (Chatain et Jard, 2004). Une extension de ce travail aux réseaux de Petri temporels est donnée dans (Jard *et al.*, 2005).

Nous retrouvons également les approches par chroniques présentées dans la section précédente mais avec des modèles de défaillances (Dousson et Maigat, 2006; Cordier *et al.*, 2007).

Les approches à base de modèles de fautes sont très puissantes en termes de détection et de diagnostic. Notamment, l'approche diagnostiqueur fournit une solution peu coûteuse en temps de calcul en ligne permettant d'assurer une bonne couverture des défaillances ayant été décrites dans une étude préalable. En ce sens le diagnostiqueur est un bon candidat pour les systèmes embarqués critiques comme dans le domaine automobile ou aéronautique.

La problématique de nos travaux porte sur le diagnostic de défaillances dans les systèmes à événements discrets mais également essentiellement sur les défaillances intermittentes. C'est pourquoi dans la section suivante nous allons faire une synthèse succincte des différentes méthodes utilisées pour le diagnostic de défaillances intermittentes, et de façon plus spécifique appliquées aux domaine des systèmes à événements discrets.

3.3 Diagnostic des défaillances intermittentes

Les premiers travaux les plus significatifs sur le diagnostic de défaillances intermittentes ont été menés dans le domaine des circuits numériques dans les années 70s. La plupart des approches (Breuer, 1973; Kamal et Page, 1974; Savir, 1980) utilise les chaînes de Markov pour modéliser les changements d'états du circuit d'un mode à l'autre à partir des données d'entrée.

Pour le diagnostic des défaillances intermittentes dans les systèmes continus, les approches à base de modèles semblent les plus adéquates. L'objectif étant pour ces approches d'estimer les paramètres du système, il est possible de détecter ces types de défaillances sans pour autant avoir un modèle de fautes. Le problème est de déterminer si l'écart observé est dû à une défaillance intermittente ou bien aux bruits de mesure par exemple. Néanmoins, les méthodes à base de données peuvent aussi être employées. Par exemple, dans (Anderson et Aylward, 1993), les réseaux de neurones sont utilisés pour leur avantage concernant leur capacité d'apprentissage de règles complexes entre différents paramètres (ici des paramètres continus). Mais comme nous l'avons vu dans le chapitre 1, il est nécessaire d'avoir une base d'apprentissage conséquente pouvant inclure des informations de défaillances. (Madden et Nolan, 1999) emploient des arbres de faute pour détecter et identifier des défaillances intermittentes. La principale difficulté pour les approches appliquées aux systèmes continus est de réussir à faire la distinction entre des mesures aberrantes provoquées par des bruits de mesures ou à des erreurs de modélisation, et la faute intermittente.

Le diagnostic des défaillances intermittentes a fait l'objet de nombreuses études et les domaines sont divers. Cependant, l'étude des défaillances fugitives reste difficile et c'est pourquoi ces travaux restent malgré tout marginaux. Pour ce qui nous intéresse, i.e. les systèmes à événements discrets, nous retrouvons différentes approches.

Dans les travaux de (Jiang *et al.*, 2003; Jiang et Kumar, 2006), les auteurs utilisent la logique linéaire temporelle introduite par (Emerson, 1990) pour faire de la détection et du diagnostic de défaillances répétitives. Ils se focalisent essentiellement sur le nombre d'occurrences des défaillances. Il n'y est pas explicitement indiqué qu'il s'agit de défaillances intermittentes mais nous pouvons y voir implicitement une disparition des défaillances comme dans (Contant *et al.*, 2004). Néanmoins, tout l'objectif de ce travail est de déterminer le nombre d'occurrences des défaillances afin de prendre en compte l'aspect dynamique du système c'est-à-dire prendre en compte le fait qu'un système ne reste pas forcément bloqué. Ainsi, au lieu de considérer le diagnostiqueur de Sampath, nous supposons que ces événements peuvent disparaître, ce qui est supposé dans les travaux de (Correcher *et al.*, 2003; Contant *et al.*, 2002; Contant *et al.*, 2004). Ces événements sont considérés comme non observables. Reprenons l'exemple de (Contant *et al.*, 2002) figure 3.3.

Nous retrouvons les événements observables ρ, α, λ, β, δ, un événement associé à une défaillance $f1$, σ_{f1} et l'événement associé à la disparition de $f1$, $\overline{\sigma_{f1}}$.

- N correspond à l'étiquette du mode normal,
- F_i correspond à l'étiquette d'une défaillance non-intermittente et présente,
- $\widetilde{F_i}$ correspond à l'étiquette de la disparition de la défaillance F_i,
- $\widetilde{\widetilde{F_i}}$ correspond à l'étiquette d'une défaillance intermittente et présente.

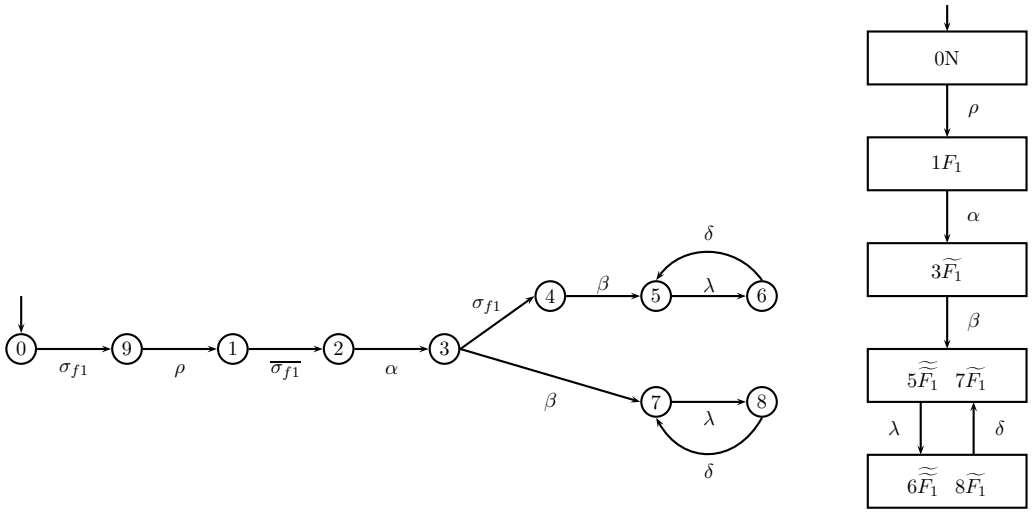


Figure 3.3 — Exemple d'un diagnostiqueur avec des défaillances intermittentes

Dans cet exemple, à l'état initial, le diagnostiqueur est dans l'état 0 et aucune défaillance ne s'est produite (label N). Si ρ est observé, le modèle se trouve dans l'état 1 correspondant à l'occurrence d'une défaillance de type f_1 ($1F_1$). Nous sommes sûrs à ce moment que la défaillance a eu lieu. Si maintenant α est observé, le modèle se trouve dans l'état 3 correspondant à l'occurrence d'un événement associé à la disparition de la défaillance de type f_1 ($3\widetilde{F_1}$). Si, à présent, l'événement β est reçu, soit nous nous trouvons dans l'état 5, correspondant au fait qu'un événement fautif f_1 a été reçu, ainsi que l'événement correspondant à la disparition de f_1, et que f_1 est toujours présent ($5\widetilde{\widetilde{F_1}}$) soit nous nous trouvons dans l'état 7 correspondant au fait que l'événement fautif f_1 a été reçu, ainsi que l'événement correspondant à la disparition de f_1, mais que f_1 n'est plus présente $7\widetilde{F_1}$. Nous sommes donc sûrs que la défaillance intermittente a eu lieu mais nous ne savons pas si la défaillance est encore présente ou non.

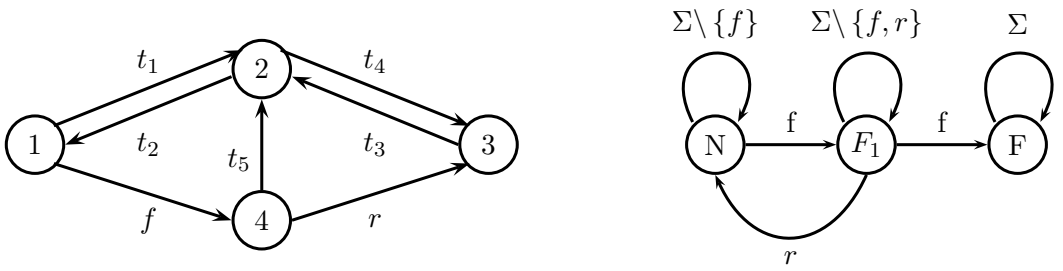
Dans cette approche, les événements de fautes sont explicitement pris en compte ainsi que les événements correspondant à leur disparition.

Dans (Jéron *et al.*, 2006), les auteurs proposent d'utiliser des modèles de supervision ("supervision pattern" en anglais). Un modèle de supervision est un automate dont le langage est l'ensemble des trajectoires à diagnostiquer. Il s'agit de déterminer les séquences d'événements correspondant aux défaillances. L'objet de ces travaux est de couvrir de manière unifiée de nombreux objectifs de diagnostic dont celui des défaillances permanentes mais également des défaillances multiples, répétitives, des séquences ordonnées d'événements ainsi que des défaillances intermittentes.

Un modèle de supervision pour les défaillances intermittentes est donné sur la figure 3.4.

- f est un événement de défaillance (non observable),
- r est un événement associé à la disparition de f (non observable),
- Σ est l'ensemble des événements.

Le modèle du système décrit les différents états de celui-ci. La figure du supervision pattern décrit le fait qu'une défaillance (occurrence de f) se produit deux fois sans action de réparation (ou de disparition, définie par l'occurrence de r). À partir de ces deux modèles, un produit



Modèle du système — Supervision pattern

Figure 3.4 — Exemple d'un modèle de supervision pour les défaillances intermittentes

synchronisé est effectué pour labelliser les états du modèle en fonction du supervision pattern. Un diagnostiqueur au sens de Sampath est ensuite construit à partir du modèle issu de ce produit synchronisé.

La question que nous nous posons ici est de savoir dans quelle mesure nous pouvons représenter sur le modèle du système ces défaillances intermittentes. En effet, toute la difficulté consiste à savoir quand elles se produisent, et par conséquent il faudrait les prendre en compte dans tous les états de notre modèle (et pas seulement dans l'état 1 comme indiqué sur la figure). En outre, dans les travaux de (Jéron *et al.*, 2006), la défaillance est connue, ce qui n'est pas forcement le cas dans nos travaux.

3.4 Synthèse et contribution de notre travail dans le diagnostic des systèmes à événements discrets

Dans le contexte de notre étude (cf. chapitre 2), nous ne considérons ni conditions opératoires, ni raisonnement temporel même si nous prendrons en compte le temps (comme chien de garde) afin de permettre le diagnostic de certaines défaillances comme l'absence de messages.

Dans le domaine automobile, il existe pour chaque constructeur quatre ou cinq gammes de véhicules, avec des versions qui sortent tous les deux trois ans pour chaque véhicule. La taille et l'évolution des systèmes et de l'électronique, les combinaisons des défaillances possibles, les réactions du système, etc. rendent impossible l'exhaustivité dans l'anticipation des défaillances. L'ensemble des approches proposées dans le diagnostic à base de modèles de fautes présuppose la détermination préalable de toutes les défaillances possibles pouvant écarter le comportement du modèle de celui attendu. Seules les défaillances connues lors de la conception peuvent être détectées et diagnostiquées. Par exemple, sur l'exemple 3.2 concernant l'approche de Sampath, nous savons que la défaillance f_1 est suivie de la séquence d'événements $[\sigma_1\sigma_2\sigma_6\sigma_3\sigma_4]$. Pour le diagnostic des défaillances intermittentes proposées par (Contant *et al.*, 2004; Jéron *et al.*, 2006), c'est le même principe. Nous savons qu'une défaillance intermittente sera suivie de telle ou telle séquence d'événements. Or, dans notre contexte, nous n'avons aucune information sur le type de défaillances (mis à part qu'elles peuvent se traduire par l'insertion ou l'absence d'un message sur le réseau de communication), sur le moment où elles peuvent apparaître, sur les réactions du système, etc. Il est donc

impossible pour nous de considérer des modèles de défaillances. La seule connaissance que nous avons est le modèle de bon comportement du système.

Dans la plupart des approches, l'événement de faute est considéré comme non observable. Or dans notre approche, l'événement fautif, c'est-à-dire un événement qui s'insère ou qui est absent, est un événement observable. Aussi, nous allons utiliser l'approche décrite dans (Pradin-Chezalviel et Valette, 1993; Valette et Kunzle, 1994) qui nous permet de détecter l'occurrence d'un événement qui n'est pas attendu par le modèle de bon comportement, ici décrit dans le formalisme des réseaux de Petri (cf. chapitre 4).

Cette méthode suffit pour détecter mais pas nécessairement pour localiser. En effet, l'événement incohérent n'est pas forcement l'événement fautif. Celui-ci a pu être reçu antérieurement mais il correspondait aux événements attendus par le modèle. L'objectif est donc de déterminer cet événement. Nos travaux se basent sur les hypothèses de l'insertion ou l'absence d'un **unique** événement.

Les approches proposées par (Pradin-Chezalviel et Valette, 1993; Jiroveanu et Boel, 2005*a*) font des recherches par raisonnement arrière pour déterminer les trajectoires susceptibles d'expliquer l'observation reçue mais prennent pas forcement en compte le fait qu'un événement puisse s'insérer ou être absent. Nous avons voulu, dans une première étape, établir une fenêtre qui correspond à une séquence d'événements conservés et qui évoluerait en fonction de la réception des événements et du marquage courant. La suppression ou l'insertion d'un événement dans cette séquence permettra de rétablir la cohérence avec une trajectoire du modèle (cf. chapitre 5). Cependant, cette méthode comme celles développées dans les travaux cités ci-dessus, demande beaucoup de calcul en ligne ce qui, pour des systèmes embarqués critiques peut poser certains problèmes.

C'est pourquoi nous avons décidé d'aborder le problème avec une approche différente. Certes le diagnostiqueur est basé sur des modèles de défaillances, mais c'est aussi une solution qui permet d'établir un diagnostic efficace avec un minimum de calcul en ligne (simple surveillance). C'est pourquoi nous avons décidé d'utiliser une approche qui s'apparente aux supervision pattern décrit dans (Jéron *et al.*, 2006). Au lieu de considérer que la défaillance peut se produire dans un état ou à un moment donné, nous allons considérer qu'elle peut arriver dans n'importe quel état du système. Nous modifions donc le modèle de bon comportement en conséquence mais à partir des données issues du bon fonctionnement puisque c'est la seule connaissance que nous avons. Nous établissons ensuite un modèle de classe de faute qui s'apparente aux modèles de supervision (cf. chapitre 6). Or, le produit synchronisé effectué pose des problèmes d'explosion d'états. Aussi, dans le chapitre 7, nous avons développé un diagnostiqueur utilisant le formalisme des réseaux de Petri qui ne nécessite pas la construction d'un produit synchronisé.

4 Détection et diagnostic de fautes intermittentes

Introduction

L'emploi de systèmes électroniques dans les systèmes embarqués augmente de façon exponentielle dans le but d'apporter sécurité et confort. Malheureusement, ces changements s'accompagnent parfois de défaillances fugitives et intermittentes qui sont difficiles à détecter (outre les symptômes fonctionnels) et à localiser puisque de par leur nature, elles ne sont plus présentes lors du diagnostic hors-ligne. Comme nous l'avons vu dans le chapitre 3, il est difficile d'avoir un modèle de ce type de défaillances, et de rendre exhaustif un tel modèle. Le problème de diagnostic de faute fugitive et intermittente est donc un problème difficile à résoudre. Il nous faut travailler sur le modèle de comportement normal du système et établir des approches basées sur ce modèle afin de s'extraire de toute nouvelle évolution des systèmes et de nouvelles apparitions de défaillances (Soldani *et al.*, 2006).

4.1 Le modèle de bon comportement

Tout d'abord, nous entendons par modèle de comportement normal, le modèle représentant le comportement prévu du système. Ce modèle comporte à la fois un mode de bon fonctionnement mais peut également contenir des modes appelés modes dégradés. Ces modes correspondent à un mode de fonctionnement du système dans lequel une défaillance est apparue.

4.1.1 Modèle global des fonctions

4.1.1.1 Les fonctions

Un système doit généralement accomplir un certain nombre de tâches. Ces tâches sont réalisées par des fonctions. Dans notre travail, nous ne considérons qu'une seule fonction à la fois. C'est cette fonction que nous allons surveiller afin de déterminer si elle est correctement exécutée par le système. Cette surveillance se fera par l'intermédiaire du modèle de bon comportement de la fonction.

Ces modèles de bon comportement sont issus des données de conception. Ces données de

conception diffèrent souvent selon les concepteurs. Par exemple, nous pouvons retrouver des modèles sous forme de state-charts (Harel, 1987) dans le domaine automobile, ou bien des données de conception propres aux industriels, et bien d'autres formes. À partir de ces données, nous allons construire un modèle à événements discrets représentant le modèle global. Comme les données de conception diffèrent selon les concepteurs, ce modèle est construit de façon manuelle. Il sera représenté par des automates ou des réseaux de Petri. Ces derniers seront préférés dans la mesure où la représentation de la synchronisation et du parallélisme est plus simple. Néanmoins, puisque nous avons décidé d'explorer plusieurs approches, nous utiliserons les deux formalismes.

4.1.1.2 Les différents formalismes utilisés

Les deux formalismes utilisés au cours de cette thèse sont les réseaux de Petri et les automates. Nous donnons ici une définition rapide de ces deux outils de représentation. Rappelons tout d'abord la définition d'un réseau de Petri.

Définition 1. ***: Réseaux de Petri***

Un réseau de Petri marqué est un 5-tuple $R =< P, T, I, O, M_0 >$*, où :*

- *P représente l'ensemble de places,*
- *T l'ensemble des transitions,*
- M_0 *est le marquage initial des places :* $M : P \rightarrow \mathbb{N}$*, où la valeur* $M_0(p)$ *est le nombre de jetons dans la place p.*
- *I est la fonction d'entrée :* $I : P \times T \rightarrow \mathbb{N}$*. La valeur I(p,t) est le poids de l'arc de la place p à la transition t.*
- *O est la fonction de sortie :* $O : P \times T \rightarrow \mathbb{N}$*, où la valeur O(p,t) est le poids de l'arc de la transition t à la place p.*

Remarque : les fonctions d'entrée I et de sortie O sont représentées sous forme matricielle. Les lignes sont associées aux places et les colonnes aux transitions.

Remarque : une transition t_i peut être tirée pour un marquage M ssi $M \geq I(., t_i)$ où $I(., t_i)$ est la i$^{\text{ième}}$ colonne de la matrice représentant la fonction d'entrée.

Nous définissons également le graphe des marquages comme étant un graphe composé de nœuds qui correspondent aux marquages accessibles, et d'arcs correspondant aux franchissements de transitions faisant passer d'un marquage à l'autre.

Exemple Sur la figure 4.1 est représenté un réseau de Petri et son graphe des marquages accessibles associé pour M_0. L'ensemble des places est $P = \{p_1, p_2\}$, l'ensemble des transitions est $T = \{t_1, t_2\}$, $M_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, et

$$I = \begin{array}{c} \\ p_1 \\ p_2 \end{array} \begin{array}{c} \begin{array}{cc} t_1 & t_2 \end{array} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array} \qquad O = \begin{array}{c} \\ p_1 \\ p_2 \end{array} \begin{array}{c} \begin{array}{cc} t_1 & t_2 \end{array} \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{array}$$

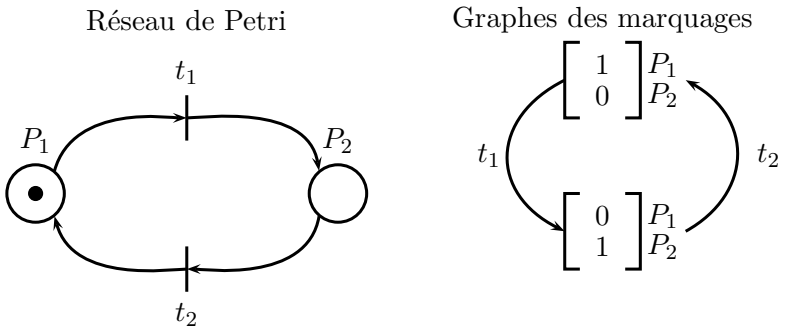


Figure 4.1 — Exemple de réseau de Petri

Définition 2. ***: Systèmes de transitions étiquetées (STE)***

Le terme d'automate est un terme générique que nous utilisons pour désigner de nombreuses variantes du même concept appelé systèmes de transitions, structures de Kripke, systèmes de transitions étiquetées, structures de Kripke étiquetées et automates. Le vocable d'automate est plus particulièrement utilisé en théorie des langages pour un système de transitions étiquetées.

Un STE est un quadruplet $\Gamma =< S, S_0, T, R >$ *où :*

- *S est l'ensemble des états ;*
- S_0 *est l'ensemble états initiaux* $S_0 \subseteq S$ *;*
- *T est un ensemble fini d'étiquettes (noms d'actions ou d'événements) ;*
- *R est l'ensemble des transitions étiquetées* $R \subseteq S \times T \times S$*.*

Remarque :
– T est l'ensemble des étiquettes des transitions ;
– Les transitions étiquetées, également appelées arcs, sont notées (s,a,s') ou $s \xrightarrow{a} s'$;
– Nous notons par ϵ l'étiquette vide telle que la transition étiquetée par celle-ci corresponde à (s,ϵ,s) ou $s \xrightarrow{\epsilon} s$ ($\epsilon \in T$).

Exemple Sur la figure 4.2 est représenté un automate. L'ensemble des états $S = \{s_1, s_2\}$, $S_0 = \{s_1\}$, $T = \{a, b\}$, et $R = \{(s_1, a, s_2), (s_2, b, s_1)\}$.

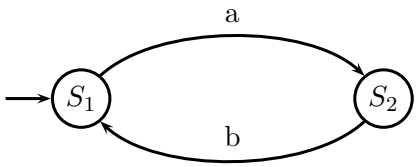


Figure 4.2 — Exemple d'automate

Pour mettre en place le modèle de bon comportement, l'utilisation de deux formalismes n'a pas été une contrainte puisque la correspondance entre les deux modèles s'est faite de

façon immédiate sans que les problèmes de synchronisme ou de parallélisme ne se posent pour les fonctions industrielles étudiées. Une correspondance existe entre les marquages M_i du réseaux de Petri et les états S_i de l'automate.

$$M_i \Longleftrightarrow S_i$$

De même, une correspondance existe entre les transitions t_i du réseau de Petri et les arcs r_i de l'automate.

$$t_i \Longleftrightarrow r_i = (s, a, s')$$

En effet, avec la correspondance marquage/état, (M_i, a, M_i') définit complètement la transition t_i puisque $I(., t_i) = M_i$ et $O(., t_i) = M_i'$. L'étiquette a correspond à l'étiquette de la transition t_i définie plus loin.

4.1.1.3 Représentation de la fonction

Pour une fonction, les marquages du modèle représentent les états de celle-ci ou des états intermédiaires. Aux marquages du modèle sont associés un label. L'ensemble des labels est noté L. Il existe autant de labels que de marquages. Il y a donc card(L)=card($\mathcal{M}$), où $\mathcal{M}$ représente l'ensemble des marquages.

Soit l, la fonction qui associe ces marquages à ces différents labels.

$$\begin{aligned} l: \quad & \mathcal{M} \rightarrow (\mathrm{L}) \\ & M \mapsto label \end{aligned}$$

Par construction, cette fonction est bijective car à chaque marquage est associé un unique label et que card(L)=card($\mathcal{M}$). Autrement dit, $\exists! M_i \in \mathrm{M} | M_i = l^{-1}(label)$.

Aux arcs du modèle sont associés des événements qui font changer le modèle d'état.

Définition 3. *Soit E l'ensemble des événements du modèle.*

Pour l'automate, ces événements correspondent en fait aux étiquettes de celui-ci. Ainsi, pour un automate, $\forall e_i \in T, e_i \in E$. Pour un réseau de Petri, il faut définir une application qui associe aux transitions un événement $e_i \in E$.

Soit l'application E_t qui associe un événement à une transition par :

$$\begin{aligned} E_t: \quad & T \rightarrow E \\ & t_i \mapsto E_t(t_i) \end{aligned}$$

où $E_t(t_i) = e_k$ avec $k \in \mathbb{N}$.

La correspondance entre les $E_t(t_i)$ et les étiquettes des automates est ainsi obtenue.

Soit l'application E_s qui associe une séquence d'événements à une séquence de transitions par :

$$E_s(s_j) = e_1 e_2 ... e_i$$

où $s_j = t_1 t_2 ... t_k$ et $E_s(s_j)[q] = E_t(s_j[q])$, où q représente le q$^{\text{ième}}$ élément.

À présent, il faut prendre en considération que les événements appartiennent à deux classes différentes : les événements observables et non observables.

Définition 4. *Notons $E_{obs} \subseteq E$ l'ensemble des événements observables du système.*

Définition 5. *Notons $E_{nobs} \subseteq E$ l'ensemble des événements non observables du système.*

Nous utilisons la notation e_{no} pour indiquer qu'un événement est non observable.

Exemple La figure 4.3 représente un réseau de Petri auquel nous avons associé des événements.

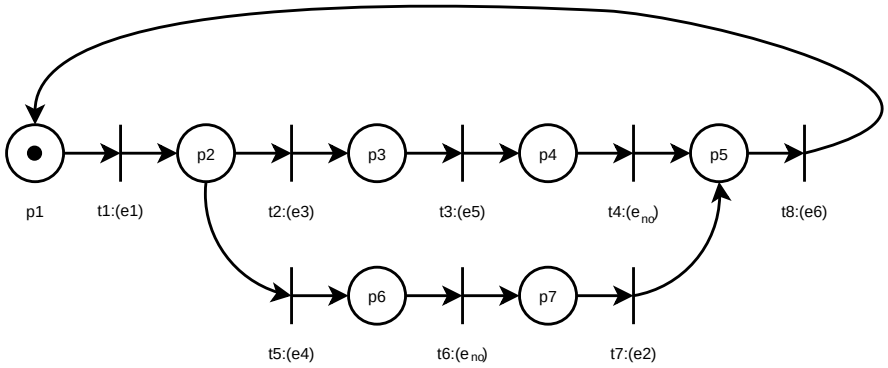


Figure 4.3 — Exemple d'un modèle de réseau de Petri global

Il est défini par :

- P=$\{p_i, i \in [1,\dots,7]\}$, T=$\{t_i, i \in [1,\dots,8]\}$, $M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{matrix}$

$$I = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{array} \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \quad O = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{array} \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Avec

$$\begin{aligned} &l(M_i) = p_i, i \in [1,\dots,7] \\ &E_t(t_1) = e_1 \quad E_t(t_5) = e_4 \\ &E_t(t_2) = e_3 \quad E_t(t_6) = e_{no} \\ &E_t(t_3) = e_5 \quad E_t(t_7) = e_2 \\ &E_t(t_4) = e_{no} \quad E_t(t_8) = e_6 \\ &E_{obs} = \{e_1, e_2, e_3, e_4, e_5, e_6\} \qquad E_{nobs} = \{e_{no}\} \end{aligned}$$

La figure 4.4 représente un automate avec les mêmes événements associés (les étiquettes) que ceux du réseau de Petri.

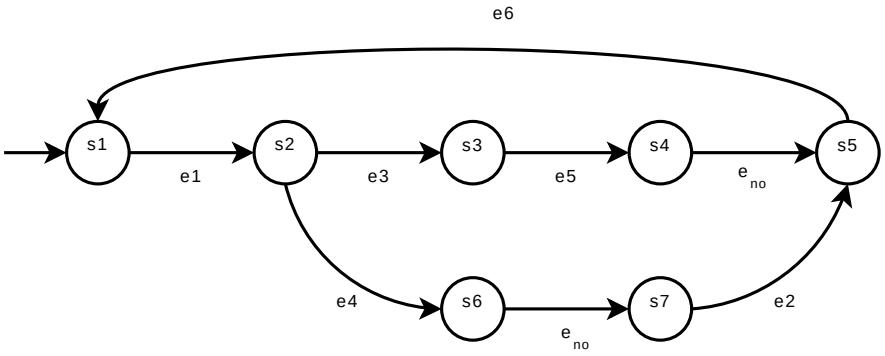


Figure 4.4 — Exemple d'un modèle automate global

Cet automate est défini par :

$$S = \{s_i\}, i \in [1, \ldots, 7] \quad T = \{e_1, e_2, e_3, e_4, e_5, e_6, e_2, e_{no}\} \quad S_0 = \{s_1\}$$
$$R = \{(s_1, e_1, s_2), (s_2, e_3, s_3), (s_3, e_5, s_4), (s_4, e_{no}, s_5), (s_2, e_4, s_6), (s_6, e_{no}, s_7), (s_7, e_2, s_5), (s_5, e_6, s_1)\}$$
$$E_{obs} = \{e_1, e_2, e_3, e_4, e_5, e_6\} \qquad E_{nobs} = \{e_{no}\}$$

En outre, à ces transitions peuvent être associées des temporisations qui permettent de prendre en compte l'aspect temporel du système. Ces temporisations seront définies sur le modèle par le concepteur ou ajoutées par la suite dans le but d'un diagnostic plus précis (cf. paragraphe 4.1.3).

Les fonctions modélisées peuvent être distribuées ou non, interagir avec plusieurs sous-systèmes. Elles font appel à un grand nombre de variables. Or ces variables ne sont pas toutes observables.

Pour prendre en considération les événements non observables, il existe plusieurs techniques. Les approches de fuzzification et de défuzzification consistent à ajouter de l'incertitude dans les modèles. Par exemple, s'il existe un doute sur le tir d'une transition, alors il existe une incertitude sur les marquages (sommes-nous toujours dans le même marquage ou dans le suivant ?). (Cardoso *et al.*, 1996) donne une vue d'ensemble des différentes techniques qui sont utilisées pour ce genre d'approches pour les systèmes à événements discrets. Dans (Cardoso *et al.*, 1999), si deux transitions sont sensibilisées et qu'il existe une incertitude (par exemple, le temps devient trop long et le marquage aurait dû changer), alors les deux transitions sont "pseudo-tirées". Des jetons sont placés dans les places correspondant aux marquages obtenus après les tirs des deux transitions. L'ensemble flou obtenu est constitué des jetons placés après les pseudo-tirs. Le marquage redevient certain lorsque le tir d'une transition est fait avec certitude. Les jetons de l'ensemble flou sont alors supprimés.

Cependant, dans nos travaux, les états qui sont liés par des événements non observables sont regroupés, comme c'est le cas pour le diagnostiqueur présenté dans le chapitre 3. Les états regroupés représentent cette ambiguïté. Conserver le modèle global et y intégrer de l'incertitude complexifierait les calculs et n'apporterait rien quant à la détection ou la localisation d'une défaillance au sens où nous l'entendons. Nous préférons donc raisonner sur un modèle plus simple et revenir sur le modèle complet dans un diagnostic hors-ligne par exemple.

Le modèle global est donc projeté sur les événements observables, à savoir les événements qui circulent sur le réseau de communication.

4.1.2 Projection du modèle sur les observables

4.1.2.1 Mécanisme d'abstraction

Le modèle est projeté sur les événements qui circulent sur le réseau de communication. Pour cela, le modèle global du comportement normal de notre fonction est modifié. Les places ou états de notre fonction qui sont reliés par un arc ou une transition auquel n'est associé aucun événement observable sont regroupés. Il est certain que notre modèle perd en précision et des "macro-états" impossible à discriminer apparaissent. Il y a là une ambiguïté voire même une perte d'information car nos travaux portent désormais sur ces modèles projetés. Cependant ces modèles étant connus, la réutilisation de la connaissance globale de la fonction permettra de faire un diagnostic plus précis. Pour la formalisation, les deux cas qui nous intéressent, à savoir celui des réseaux de Petri et des automates seront étudiés.

Automates Nous rappellons les définitions de partition et d'automate quotient que nous retrouvons dans (Dupont et Miclet, 1998) :

Définition 6. *Pour tout ensemble A, une partition π est un ensemble de sous ensembles de A, non vides et disjoints deux à deux, dont l'union est A. Si a désigne un élément de A, B(a, π) désigne l'unique élément de π comprenant a.*

Définition 7. *Si $\Gamma =< S, S_0, T, R >$ est un automate, l'automate $\Gamma/\pi =< S', B(S_0, \pi),$- $T, R' >$ dérivé de Γ relativement à la partition π de S, appelé aussi l'automate quotient Γ/π, est défini comme suit :*
- $S' = S/\pi = \{B(s, \pi)|s \in S\}$,
- $R' : S' \times T \times S' : \forall B, B' \in S', \forall t \in T, (B, t, B') \in R'$ si et seulement si $\exists q, q' \in S, q \in B, q' \in B'$ et $(q, t, q') \in R$.

S' et T sont des ensembles finis, et S_0 un ensemble d'état initiaux. R' est un ensemble de transitions étiquetées tel que $R' \subseteq S' \times T \times S'$. C'est donc un automate.

Pour projeter le modèle sur les observables, il faut construire la partition π. Pour cela, il suffit de rassembler les états qui ne sont reliés par aucun événement observable. En d'autres termes, S' est construit initialement comme étant l'ensemble des singletons de S : $\pi = \{\{S_i\}\}, \forall S_i \in S$. π est bien une partition de S. Construisons S' en prenant en compte les contraintes.
$Si\ \forall r = (q, t, q') \in R | t = e_i\ et\ e_i \notin E_{obs}\ alors\ \pi = \pi \cup \{\{B(q, \pi) \cup B(q', \pi)\}\}$ $\backslash\{\{B(q, \pi)\}, \{B(q', \pi)\}\}$.

Une nouvelle partition S' de notre ensemble d'états est obtenue. Cette partition permet de construire le modèle abstrait par fusion d'états à l'aide de la définition 7. Le modèle obtenu est bien, par construction, un automate.

Cependant, en appliquant cette fusion d'états, le modèle obtenu conserve encore les arcs dont les éléments sont non observables. Comme nous ne voulons conserver sur le modèle que

les arcs qui ont des éléments observables, ces arcs sont supprimés. Par conséquent, $si\ \forall r' = (s,t,s') \in R' | t = e_i \dots e_i \notin E_{obs}\ alors\ R' = R' \backslash \{r'\}$. Cela n'a aucune incidence sur la structure de l'automate dans la mesure où par construction $s = s'$ et donc seul un arc qui a pour état de sortie son état d'entrée est supprimé. Finalement, le résultat reste encore un automate.

Exemple En reprenant l'exemple précédant de la figure 4.4, nous allons procéder à la projection de ce modèle sur nos observables i.e. E_{obs}.

Dans un premier temps, construisons la partition π. Initialement, $\pi = \{\{s_i\}\}, i \in [1,\dots,7]$. Il existe deux arcs $r = (q,t,q') \in R$ tels que $t = e_i\ et\ e_i \notin E_{obs}$. Il s'agit de : $\{(s_4, e_{no}, s_5), (s_6, e_{no}, s_7)\}$.

Alors la nouvelle partition est :
$\pi = \pi \cup \{\{B(s_4,\pi) \cup B(s_5,\pi)\}\} \backslash \{\{B(s_4,\pi)\}, \{B(s_5,\pi)\}\}$.
$\pi = \pi \cup \{\{s_4, s_5\}\} \backslash \{\{s_4\}, \{s_5\}\}$.
$\pi = \{\{s_1\}, \{s_2\}, \{s_3\}, \{s_4, s_5\}, \{s_6\}, \{s_7\}\}$.

La même chose est faite pour $r = (s_6, e_{no}, s_7)$. Ce qui donne :
$\pi = \{\{s_1\}, \{s_2\}, \{s_3\}, \{s_4, s_5\}, \{s_6, s_7\}\}$.

L'automate quotient obtenu est défini par (cf. def. 7) :
- $S' = S/\pi = \{B(s,\pi) | s \in S\} = \{\{s_1\}, \{s_2\}, \{s_3\}, \{s_4, s_5\}, \{s_6, s_7\}\}$,
- $R' = \{(\{s_1\}, e_1, \{s_2\}), (\{s_2\}, e_3, \{s_3\}), (\{s_3\}, e_5, \{s_4, s_5\}), (\{s_2\}, e_4, \{s_6, s_7\}),$
$(\{s_6, s_7\}, e_2, \{s_4, s_5\}), (\{s_4, s_5\}, e_6, \{s_1\}), (\{s_4, s_5\}, e_{no}, \{s_4, s_5\}), (\{s_6, s_7\}, e_{no}, \{s_6, s_7\})\}$
Il ne reste plus qu'à supprimer les arcs dont tous les éléments sont non observables. Soit :
$R' = \{(\{s_1\}, e_1, \{s_2\}), (\{s_2\}, e_3, \{s_3\}), (\{s_3\}, e_5, \{s_4, s_5\}), (\{s_2\}, e_4, \{s_6, s_7\}),$
$(\{s_6, s_7\}, e_2, \{s_4, s_5\}), (\{s_4, s_5\}, e_6, \{s_1\})\}$
Le modèle obtenu est représenté sur la figure 4.5.

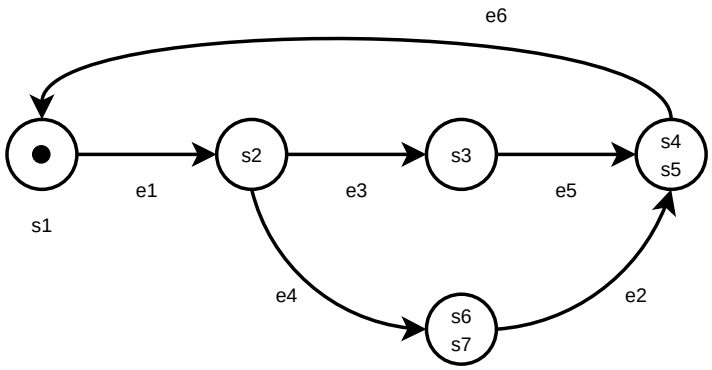


Figure 4.5 — Modèle abstrait de l'automate

Réseau de Petri

Définition 8. *Si $R =< P, T, I, O, M_0 >$ est un réseau de Petri, le réseau de Petri $R/\pi =< P', T, I', O', M'_0 >$ dérivé de R relativement à la partition π de P, appelé aussi réseau de Petri*

quotient R/π, est défini comme suit :
- $P' = P/\pi = \{B(p,\pi)|p \in P\}$,
- I' a pour dimension $dim(I') = P' \times T$ et $\forall B \in \pi, \forall p \in B, I'(B(p,\pi),.) = \sum_p I(p,.)$,
- O' a pour dimension $dim(O') = P' \times T$ et $\forall B \in \pi, \forall p \in B, O'(B(p,\pi),.) = \sum_p O(p,.)$, -
M'_0 est défini tel que $\forall p \in P|M_0(p) = 1$ alors $M'_0(B(p,\pi)) = 1$.

P' et T sont deux ensembles finis. I' et O' sont des fonctions de $P' \times T \rightarrow \mathbb{N}$. M'_0 est un marquage tel que $M'_0 : P' \rightarrow \mathbb{N}$. C'est donc bien un réseau de Petri.

Comme pour les automates, nous devons construire notre partition π. Pour cela, il suffit de rassembler les places qui sont reliées par des transitions dont aucun événement associé n'est observable. En d'autres termes, P' est construit initialement comme étant l'ensemble des singletons de P : $\pi = \{\{p_i\}\}, \forall p_i \in P$. π est bien une partition de P. Construisons P' en prenant en compte les contraintes.

$\forall t \in T$ *si* $e \in E_t(t), e \notin E_{obs}$ *alors* $\pi = \pi \cup \{\bigcup_p B(p,\pi)\} \backslash \{\{B(p,\pi)\}\}, \forall p \in P|I(p,t) = 1$ *ou* $O(p,t) = 1$.

Une nouvelle partition P' de notre ensemble de places est construite. Cette partition permet de construire le modèle abstrait à l'aide de la définition 8.

Nous devons à présent faire une remarque quant à la fusion des places dans les réseaux de Petri. Deux places liées par une transition t_i ne peuvent être fusionnées si la transition a plus d'une place en entrée ou plus d'une place en sortie (voir figure 4.6). Cette transition non observable doit être conservée pour que les propriétés du réseau de Petri restent satisfaites.

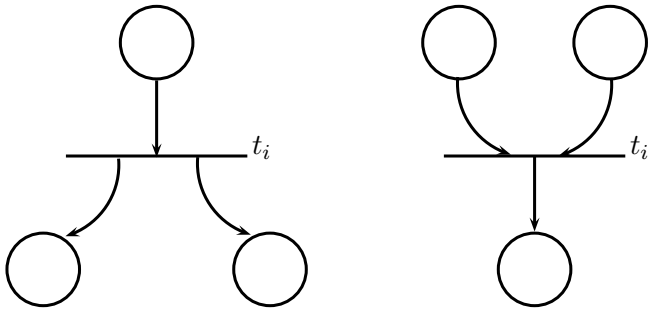


Figure 4.6 — Fusion : problème de synchronisation et de parallélisme

Avec la fusion des places dans les réseaux de Petri définie précédemment, le modèle obtenu conserve encore les transitions dont les éléments sont non observables. Comme nous ne voulons conserver sur notre modèle que les transitions qui ont des éléments observables, ces transitions sont supprimées. Par conséquent, $\forall t \in T$ *si* $e \in E_t(t), e \notin E_{obs}$ *alors* $T = T \backslash \{t\}, I'_{new} = I' \backslash I'(.,t), O'_{new} = O' \backslash O'(.,t)$ (les colonnes correspondant à la transition t sont supprimées). Cela n'a aucune incidence sur la structure du réseau de Petri dans la mesure où par construction $I'(.,t) = O'(.,t)$, c'est-à-dire que la transition qui a ses places de sortie identiques à ses places d'entrée est supprimée. Soit T' le nouvel ensemble de transitions. Ce nouvel ensemble de transitions T' reste un ensemble fini. Les deux fonctions I' et O' sont des fonctions de $P' \times T' \rightarrow \mathbb{N}$. Le résultat est bien au final un réseau de Petri.

Exemple En reprenant l'exemple précédant de la figure 4.3, nous allons procéder à la projection de ce modèle sur les observables i.e. E_{obs}.

Comme pour l'automate, la partition π est construite. Initialement, $\pi = \{\{p_i\}\}, i \in [1, \ldots, 7]$. Il existe seulement deux transitions telles que $\forall e \in E_t(t), e \notin E_{obs}$. Il s'agit de t_4 et t_6.

Pour t_4, seuls $I(p_4, t_4) = 1$ et $O(p_5, t_4) = 1$.
$\pi = \pi \cup \{\{B(p_4, \pi) \cup B(p_5, \pi)\}\}\backslash\{\{B(p_4, \pi)\}, \{B(p_5, \pi)\}\}$.
$\pi = \pi \cup \{\{p_4, p_5\}\}\backslash\{\{p_4\}, \{p_5\}\}$.
$\pi = \{\{p_1\}, \{p_2\}, \{p_3\}, \{p_4, p_5\}, \{p_6\}, \{p_7\}\}$.

De la même manière, pour t_6, seuls $I(p_6, t_6) = 1$ et $O(p_7, t_6) = 1$. Finalement, il en résulte :
$\pi = \{\{p_1\}, \{p_2\}, \{p_3\}, \{p_4, p_5\}, \{p_6, p_7\}\}$.

Une nouvelle partition est construite. À présent, à partir de la définition 8, le réseau de Petri projeté sur les observables peut être construit.
- $P' = P/\pi = \{B(p, \pi) | p \in P\} = \{\{p_1\}, \{p_2\}, \{p_3\}, \{p_4, p_5\}, \{p_6, p_7\}\}$,
- $I'(\{p_1\}, .) = I(p_1, .)$, $I'(\{p_2\}, .) = I(p_2, .)$, $I'(\{p_3\}, .) = I(p_3, .)$, $I'(\{p_4, p_5\}, .) = I(p_4, .) + I(p_5, .)$, $I'(\{p_6, p_7\}, .) = I(p_6, .) + I(p_7, .)$

$$I' = \begin{array}{c} \\ \{p_1\} \\ \{p_2\} \\ \{p_3\} \\ \{p_4,p_5\} \\ \{p_6,p_7\} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 \end{array} \\ \left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right] \end{array}$$

- $O'(\{p_1\}, .) = O(p_1, .)$, $O'(\{p_2\}, .) = O(p_2, .)$, $O'(\{p_3\}, .) = O(p_3, .)$, $O'(\{p_4, p_5\}, .) = O(p_4, .) + O(p_5, .)$, $O'(\{p_6, p_7\}, .) = O(p_6, .) + O(p_7, .)$

$$O' = \begin{array}{c} \\ \{p_1\} \\ \{p_2\} \\ \{p_3\} \\ \{p_4,p_5\} \\ \{p_6,p_7\} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 \end{array} \\ \left[\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right] \end{array}$$

À présent il faut supprimer les colonnes qui correspondent aux transitions non observables, i.e. t_4 et t_6. Finalement, cela donne :

$$I' = \begin{array}{c} \\ \{p_1\} \\ \{p_2\} \\ \{p_3\} \\ \{p_4,p_5\} \\ \{p_6,p_7\} \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & t_3 & t_5 & t_7 & t_8 \end{array} \\ \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \end{array} \quad O' = \begin{array}{c} \\ \{p_1\} \\ \{p_2\} \\ \{p_3\} \\ \{p_4,p_5\} \\ \{p_6,p_7\} \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & t_3 & t_5 & t_7 & t_8 \end{array} \\ \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \end{array}$$

Finalement, le réseau de Petri obtenu est représenté sur la figure 4.7.

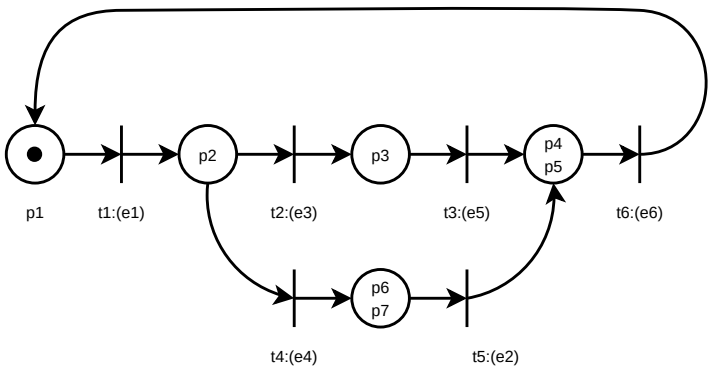


Figure 4.7 — Modèle abstrait pour le réseau de Petri

Une remarque peut déjà être faite ici quant à la construction du modèle. Les variables associées aux transitions mais également celles qui sont incluses dans les regroupements des places du modèle global doivent être remises en cause même si, sur le modèle, ne sont représentées que les variables observables.

Finalement, un modèle à événements discrets est obtenu et décrit les évolutions observables d'une fonction donnée, distribuée sur les composants d'un réseau. À partir de ce modèle, le comportement de la fonction modélisée peut être observé en suivant les messages qui circulent sur le réseau de communication.

4.1.3 Aspects temporels

Dans nos travaux, les aspects temporels ne sont pas pris en compte sur le modèle. Or, dans de nombreux cas, ces aspects doivent être pris en compte pour décrire le fonctionnement du système et apporter des informations supplémentaires. Aussi, nous présentons ici des éléments qui pourront servir par la suite pour une étude intégrant cette notion de temps. Les idées générales sont exposées sans entrer dans les détails de la formalisation ou encore des règles d'évolution du réseau de Petri.

Pour prendre en compte les aspects temporels, nous pouvons utiliser les travaux développés dans l'ouvrage (Diaz, 2001) qui reprend en partie les études menées par (Merlin, 1974). Ils définissent les réseaux de Petri temporels comme étant des réseaux de Petri dans lesquels deux valeurs min et max (avec $0 \leq min \leq max$, min fini et max éventuellement infini) sont associés à chaque transition du réseau. Ces valeurs, pour une transition t du réseau, sont relatives à la date à laquelle la transition t a été sensibilisée pour la dernière fois.

Supposons qu'à une date θ la transition t devienne sensibilisée ; la transition t ne peut alors être tirée avant la date $\theta + min$ et doit l'être au plus tard à la date $\theta + max$ (si max est fini), sauf si le tir d'une autre transition désensibilise la transition t avant que celle-ci ne soit tirée. Le tir des transitions est de durée nulle. Une fois le tir de la transition effectué, les transitions qui sont restées sensibilisées voient leur intervalle de tir décalé, vers l'origine du temps, de la valeur θ, date relative à laquelle la transition a été tirée.

L'intégration du temps sera nécessaire par la suite pour prendre en compte le fait qu'un événement est absent par exemple, et qu'il ne faut pas rester dans le même marquage indéfiniment. Prendre en compte le temps dans le modèle permet également de détecter qu'il y a une défaillance si les délais ne sont pas respectés même si l'événement arrive plus tard.

4.2 Surveillance et détection

Une fois le modèle construit, nous allons pouvoir suivre l'évolution du comportement de la fonction par l'intermédiaire des messages qui circulent sur le réseau de communication.

4.2.1 Détection des fautes intermittentes et fugitives

L'évolution de l'état du système est suivie par l'évolution du système à événements discrets à chaque occurrence d'un événement observable. L'état initial est supposé connu.

Pour chaque marquage ou état, un ensemble d'événements attendus est déterminé. Cet ensemble décrit les évolutions normales du système.

Quand un événement observable apparaît, soit il appartient à l'ensemble des événements attendus et le système à événements discrets est mis à jour c'est l'évolution normale ; soit il n'appartient pas à cet ensemble et par conséquent, une incohérence est détectée, c'est le symptôme d'une défaillance (voir figure 4.8).

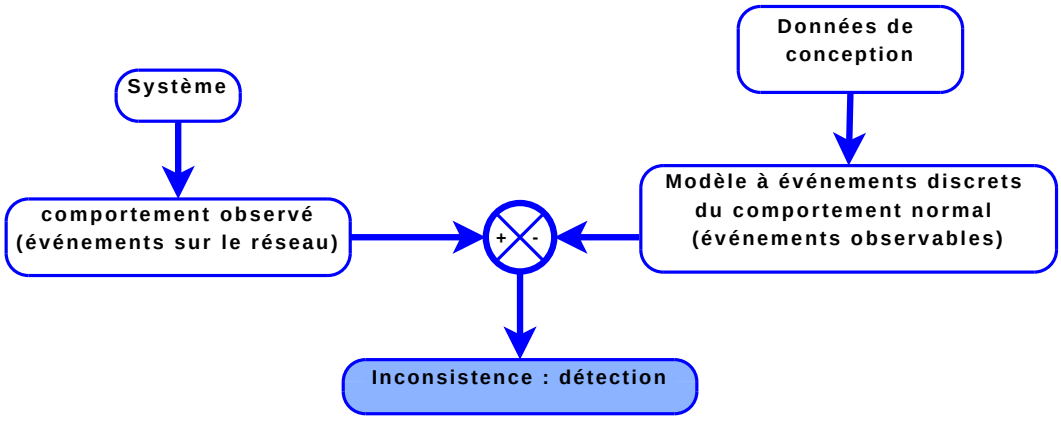


Figure 4.8 — Schéma de détection

4.2.1.1 Définitions

Soit E, l'ensemble des événements observables (def. 3) que nous notons :

$$E_{obs} = \{e_1, e_2, ..., e_n\}$$

Soit $\mathcal{P}$(X) l'ensemble des sous ensembles de X.

Définition 9. *L'application E_{TFT} qui associe à un état un ensemble d'arcs est donnée par :*

$$\begin{aligned} E_{TFT} : \quad & S \rightarrow \mathcal{P}(R) \\ & s_i \mapsto E_{TFT}(s_i) = \{r_i / r_i = (s_i, t_i, s'_i) \in R\} \end{aligned}$$

Dans le cas d'un réseau de Petri, un marquage correspond à S_i et une transition t_i à r_i telle que :

$$E_{TFT}(M_i) = \{t_i / M_i \geq I(., t_i)\}$$

Il s'agit des transitions sensibilisées.

Définition 10. *L'ensemble E_{exp} des événements associés à une transition sensibilisée à partir d'un état, i.e. l'ensemble des événements attendus, est défini comme :*

$$E_{exp}(s_i) = \{t_i | r_i = (s_i, t_i, s'_i) \in E_{TFT}(s_i)\}$$

Il s'agit des étiquettes de l'automate. Par correspondance, pour un réseau de Petri, cela donne :

$$E_{exp}(M_i) = \{E_t(t_i) | t_i \in E_{TFT}(M_i)\}$$

Définition 11. *L'application E_{TFE} qui associe un ensemble d'un ou plusieurs arcs à un événement est donnée par :*

$$\begin{aligned} E_{TFE} : \quad & S \rightarrow \mathcal{P}(R) \\ & e_i \mapsto E_{TFE}(e_i) = \{r_i / r_i = (s_i, e_i, s'_i) \in R\} \end{aligned}$$

Par correspondance, pour un réseau de Petri, il en résulte :

$$E_{TFE}(e_i) = \{t_i / e_i \in E_t(t_i)\}$$

Définition 12. *Soit un état S_i et l'occurrence d'un événement e_j, l'ensemble $E_F(S_i, e_j)$ des arcs qui peuvent faire évoluer le modèle est :*

$$E_F(S_i, e_j) = E_{TFE}(e_j) \bigcap E_{TFT}(S_i)$$

4.2.1.2 Principe de détection

Le suivi du comportement se fait donc par rapport à l'événement reçu et aux événements attendus.

- Si $E_F(S_i, e_j) = \{t_k\} \neq \emptyset$, alors le nouvel état courant S'_i est tel que $(S_i, t_k, S'_i) \in R$. Le nouveau marquage $M' = M_i - I(., t_k) + O(., t_k)$ est atteint.
- Si $E_F(S_i, e_j) = \emptyset$, alors cela signifie qu'il existe une incohérence entre l'événement reçu et l'ensemble des événements attendus. Il s'agit du principe de détection.

Cette incohérence peut avoir différentes causes :

- un événement reçu s'est inséré de façon inopinée,

– un événement est absent,
– un arc est manquant dans le système à événements discrets. C'est un problème de modélisation. Dans nos travaux, nous ne considérons pas ce cas là mais nous pouvons nous reporter aux travaux effectués dans la thèse de Carmen Lopez Varela (Lopez-Varela, 2007) ou dans les travaux de (Yeung et Kwong, 2005) et (Whiteford et Kwong, 2007).

Exemple Reprenons l'exemple du réseau de Petri représenté sur la figure 4.7. L'approche avec l'automate étant similaire, seul le cas du réseau de Petri est présenté.

Le marquage initial est $M_0 = \begin{bmatrix} 1 & p_1 \\ 0 & p_2 \\ 0 & p_3 \\ 0 & \{p_4, p_5\} \\ 0 & \{p_6, p_7\} \end{bmatrix}$

Les événements associés aux transitions sont $E_t(t_1) = \{e_1\}$, $E_t(t_2) = \{e_3\}$, $E_t(t_3) = \{e_5\}$, $E_t(t_4) = \{e_4\}$, $E_t(t_5) = \{e_2\}$, $E_t(t_6) = \{e_6\}$.

Les transitions associées aux événements sont $E_{TFE}(e_1) = \{t_1\}$, $E_{TFE}(e_2) = \{t_5\}$, $E_{TFE}(e_3) = \{t_2\}$, $E_{TFE}(e_4) = \{t_4\}$, $E_{TFE}(e_5) = \{t_3\}$, $E_{TFE}(e_6) = \{t_6\}$.

La transition sensibilisée est $E_{TFT}(M_0) = \{t_i / M_0 \geq I(., t_i)\} = \{t_1\}$. Par conséquent les événements attendus sont $E_{exp}(M_0) = \{e_1\}$.

Si le premier événement reçu est e_1 alors l'ensemble des transitions associées à e_1 est $E_{TFE}(e_1) = \{t_1\}$. Ainsi, $E_{TFT}(M_0) \bigcap E_{TFE}(e_1) \neq \emptyset = \{t_1\}$.

Il n'y a donc pas d'incohérence. Le nouveau marquage est $M_1 = \begin{bmatrix} 0 & p_1 \\ 1 & p_2 \\ 0 & p_3 \\ 0 & \{p_4, p_5\} \\ 0 & \{p_6, p_7\} \end{bmatrix}$

De la même manière, les nouvelles transitions sensibilisées et les nouveaux événements attendus sont $E_{TFT}(M_1) = \{t_2, t_4\}$ et $E_{exp}(M_1) = \{e_3, e_4\}$ respectivement.

Maintenant, supposons que le 2$^{\text{ième}}$ événement reçu est e_3. L'ensemble des transitions associé à e_3 est $E_{TFE}(e_3) = \{t_2\}$. Ainsi, $E_{TFT}(M_1) \bigcap E_{TFE}(e_3) \neq \emptyset = \{t_2\}$.

Il n'y a donc pas d'incohérence. Le nouveau marquage est $M_2 = \begin{bmatrix} 0 & p_1 \\ 0 & p_2 \\ 1 & p_3 \\ 0 & \{p_4, p_5\} \\ 0 & \{p_6, p_7\} \end{bmatrix}$

Les nouvelles transitions sensibilisées et les nouveaux événements attendus sont $E_{TFT}(M_2) = \{t_3\}$ et $E_{exp}(M_2) = \{e_5\}$ respectivement.

Maintenant, supposons que le 3$^{\text{ième}}$ événement reçu est e_4. L'ensemble des transitions associé à e_4 est $E_{TFE}(e_4) = \{t_4\}$. Ainsi, $E_{TFT}(M_2) \bigcap E_{TFE}(e_4) = \{t_3\} \bigcap \{t_4\} = \emptyset$. Il y a

donc une incohérence et le symptôme d'une défaillance est détecté.

Une fois qu'une incohérence est détectée, l'objectif est de déterminer quel est l'événement réellement en cause. En effet, l'événement détecté n'est pas forcément l'événement inséré ou absent (voir figure 4.9). Sur cet exemple, l'événement e_1 est supposé s'être inséré en p_0 et la séquence initiale d'événements $[e_2, e_3, e_5]$ est reçue. Le marquage $[000001]^T$ est atteint i.e. celui qui correspond à la place p_5. À partir de ce marquage, l'événement e_4 est attendu mais c'est l'événement e_5 qui est reçu. Il y a donc incohérence et par conséquent détection. L'événement détecté est donc e_5 mais l'événement fautif est e_1. Il faut donc élargir l'ensemble des événements mis en cause. La détermination de l'événement fautif est ce qui est considéré comme étant l'étape de localisation. C'est ce que nous allons voir dans les chapitres suivants.

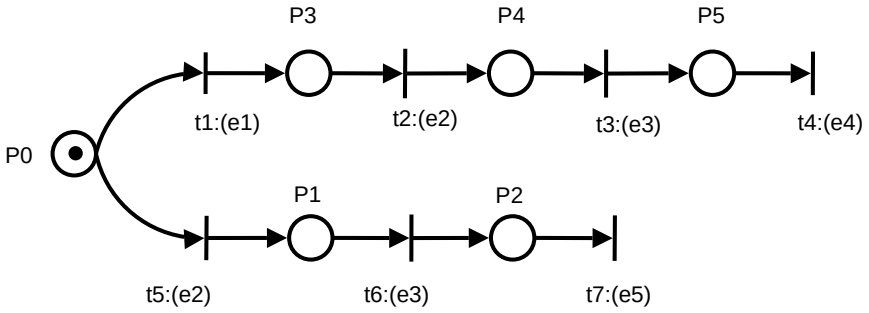


Figure 4.9 — Différence entre événement détecté et événement fautif

4.3 Réinitialisation

Avant de commencer l'étape de localisation, nous allons tout d'abord dire deux mots sur l'étape de réinitialisation. En effet, lorsque qu'une incohérence est détectée entre les observations et le modèle, cela signifie que le modèle n'est plus dans l'état qui correspond à l'état réel du système. Le modèle a suivi une trajectoire qui ne correspond plus à l'état réel du système. Le modèle est alors remis dans son état initial. Par conséquent, lors de cette phase, les événements reçus ne correspondent pas aux événements attendus et ne sont donc pas considérés comme des événements fautifs. Il faut attendre qu'un événement soit à nouveau cohérent pour faire, à nouveau, de la détection. Néanmoins, sous nos hypothèses, la détermination de l'événement absent ou inséré permet de retrouver l'état dans lequel se trouve le système.

Nous pourrons également par la suite se pencher sur les travaux, entre autres, de (Caines *et al.*, 1988) et de (Giua, 1997) pour approfondir ces aspects de réinitialisation et de la même manière, les aspects d'initialisation. En effet, une étape transitoire peut être imaginée dans laquelle l'objectif est, dans un premier temps, de déterminer l'état courant du système et seulement ensuite, recommencer à surveiller le système.

Nous pourrons aussi nous reporter aux travaux de (Qiu et Kumar, 2007) qui se penchent sur l'estimation d'état d'un système surveillé par un ensemble d'observateurs ainsi que sur les travaux de (Cabasino *et al.*, 2007) qui portent sur l'estimation de marquage de réseaux de Petri dont les transitions peuvent être non labellisées, appelées transitions non observables,

ou dont plusieurs transitions sensibilisées peuvent avoir le même label.

4.4 conclusion

Dans ce chapitre, le modèle comportemental des fonctions utilisées est représenté. Ce modèle à événements discrets est décrit soit sous la forme d'un automate soit sous la forme d'un réseau de Petri. Il représente le comportement normal du modèle global de la fonction étudiée. Il est ensuite projeté sur les observables, à savoir les messages qui circulent sur un réseau de communication. C'est à partir de ce modèle projeté que le comportement du système est surveillé en comparant les observables aux événements attendus par le modèle. Dès que le comportement sort de son comportement prévu alors cela signifie que le système est sorti de son fonctionnement normal et par conséquent qu'il existe une défaillance. Ce chapitre a permis de formaliser le mécanisme d'abstraction et le principe de détection pour les deux modèles utilisés (automates et réseau de Petri). Nous avons vu que le mécanisme de détection n'est pas suffisant dans la mesure où l'événement détecté ne correspond pas forcément à l'événement fautif (même si cela se trouve être la majorité des cas). Il faut donc établir une approche qui permette de localiser cet événement fautif dans le cas où le modèle aurait suivi une mauvaise trajectoire du modèle avant de détecter l'incohérence.

5 Approche par glissement de fenêtre

Introduction

Dans le chapitre précédant, nous avons présenté le principe de détection qui consiste à détecter une incohérence entre l'événement reçu et les événements attendus par le modèle. Mais l'événement détecté ne correspond pas forcément à l'événement fautif (l'événement qui s'est inséré ou qui est absent). Par hypothèse, nous supposons qu'un seul événement s'est inséré ou est absent, celui qui fait changer de trajectoire. Il n'est donc possible de changer de trajectoire qu'une seule fois. Nous supposons également que la détection de cet événement a lieu avant l'apparition d'une nouvelle défaillance et donc que les événements reçus par la suite correspondent encore au bon fonctionnement du système.
Il faut donc mettre en place une approche qui permette d'identifier cet événement. Il s'agit de la phase de localisation. En effet, la recherche de l'événement fautif permettra de définir un ensemble de composants susceptibles d'être à l'origine de l'émission non prévue de cet événement (ou de la non émission d'un événement prévu). Sur le schéma du diagnostic à base de modèle, la phase de localisation vient à la suite de celle de détection (cf. fig. 5.1). Dans les trois prochains chapitres, différentes méthodes seront proposées qui permettent de déterminer l'événement qui s'est inséré ou qui est absent.
La première méthode consiste à conserver une séquence des derniers événements reçus et, une fois que la détection a eu lieu, de trouver toutes les modifications qu'il faut apporter pour rétablir la cohérence entre la séquence de transitions (associée à cette séquence modifiée) et une trajectoire du modèle. La modification apportée constitue l'étape de localisation. Un modèle à base de réseaux de Petri a été utilisé dans toute cette approche.
Dans un premier temps, il faut déterminer le nombre d'événements à conserver à partir d'un marquage pour confirmer que l'événement reçu à ce marquage correspond bien à un événement issu du bon comportement.
Ensuite, en-ligne, lors de la surveillance du système, il faut calculer, de façon dynamique, le nombre d'événements qu'il faut effectivement conserver, car ce nombre dépend des états par lesquels le modèle est passé.
Ainsi, lorsqu'une détection est faite, une séquence d'événements a été conservée dans laquelle un événement a pu s'insérer ou bien être absent, et qui pourrait expliquer l'incohérence observée. Nous proposons donc une méthode exécutée en ligne et qui permet de déterminer tous les événements insérés ou absents.

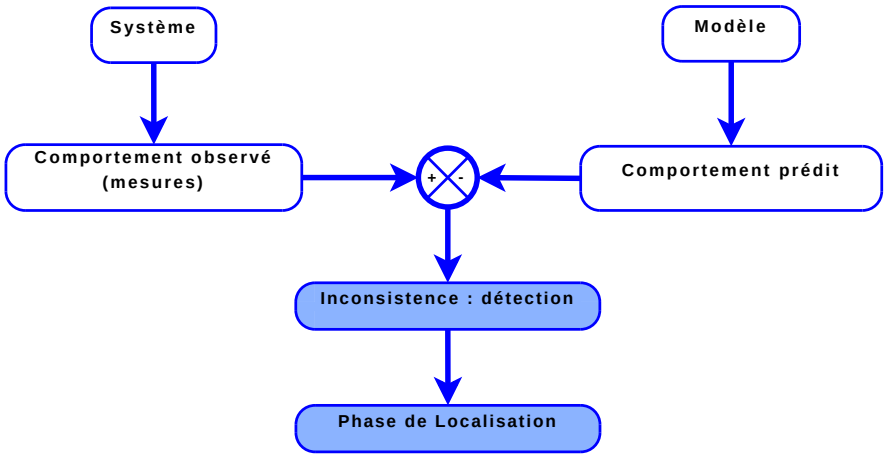


Figure 5.1 — Principe de l'approche basée modèle

5.1 Longueur de fenêtre

Lors de la surveillance du système, des événements sont reçus en séquence qui font changer le système d'état. Le système est supposé soumis à une défaillance qui entraîne l'émission d'un message (ou non) sur le réseau de communication. La détection de cette défaillance peut ne pas se faire immédiatement. C'est pourquoi, il est nécessaire de conserver la séquence des derniers événements reçus afin d'être sûr de retrouver l'événement fautif. Pour cela, il existe deux phases : la première permet de calculer le nombre d'événements à conserver à partir d'un marquage (phase hors-ligne) ; la seconde permet de déterminer le nombre d'événements à conserver réellement en fonction des marquages atteints précédemment (phase en-ligne).

5.1.1 Détermination de la longueur de la séquence à partir du marquage courant

La question posée ici est de savoir si, à partir d'un marquage certain M_i du modèle et d'une séquence d'événements reçus, la "bonne" trajectoire est suivie. Cela revient à se demander si, à partir d'un marquage certain M_i, le modèle ne va pas suivre une trajectoire différente lorsqu'un événement s'insère ou est absent. Si tel est le cas, la détection ne se fera pas immédiatement. Il est donc nécessaire de déterminer la longueur de la séquence d'événements à conserver de manière à ce que, au moment de la détection, le résultat de la défaillance (l'insertion ou l'absence d'un événement) soit toujours dans la séquence conservée. Il s'agit donc de déterminer le nombre d'événements nécessaires pour pouvoir détecter une incohérence. La réception de la totalité de ces événements à partir d'un marquage certain M_i permettra d'en déduire que l'événement reçu au marquage M_i ne provient pas d'une défaillance et donc, que le marquage suivant M_{i+1} est certain. Cela n'est vrai que sous l'hypothèse qu'un seul événement est absent ou s'est inséré.
En reprenant l'exemple (fig. 5.4), à partir du marquage $M_0 = [10000]^T$, deux trajectoires sont possibles. Celle qui correspond à la séquence $s_1 = [e_1, e_2, e_3, e_4]$ et celle qui correspond

à la séquence $s_2 = [e_2, e_3, e_5]$. Nous supposons que la séquence s_2 doit être reçue mais que lorsque le marquage M_0 est atteint, l'événement e_1 s'insère. La séquence $s = [e_1, e_2, e_3, e_5]$ est finalement reçue. Il faut attendre le $4^{\text{ième}}$ événement afin de pouvoir discriminer les deux trajectoires. La réception de l'événement e_5 confirme que l'insertion de l'événement e_1 est une hypothèse valide permettant d'expliquer les observations reçues. Si e_4 avait été reçu, cela aurait confirmé que l'événement e_1 n'était pas le produit d'une défaillance et par conséquent que le marquage suivant $M_1 = [00100]^T$ est certain.
Considérer l'insertion de e_1 dans la séquence s_2 ou son absence dans la séquence s_1 change le nombre d'événements qui doit être conservé. Comme, ensuite, un travail est effectué sur cette séquence pour localiser un événement inséré ou absent, il faut conserver le plus grand nombre d'événements, ce qui correspond à l'insertion d'un événement (cf. figure 5.2 et 5.3). Dans le cas de l'insertion, il faut conserver quatre événements pour détecter l'incohérence alors que dans le cas de l'absence, il suffit d'en conserver trois. Nous sommes sûr que l'insertion ou l'absence de l'événement s'est produit lors de la réception de ces événements.

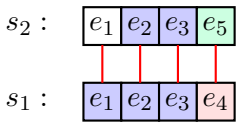


Figure 5.2 — Cas de l'insertion de e_1

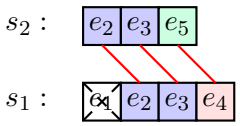


Figure 5.3 — Cas de l'absence de e_1

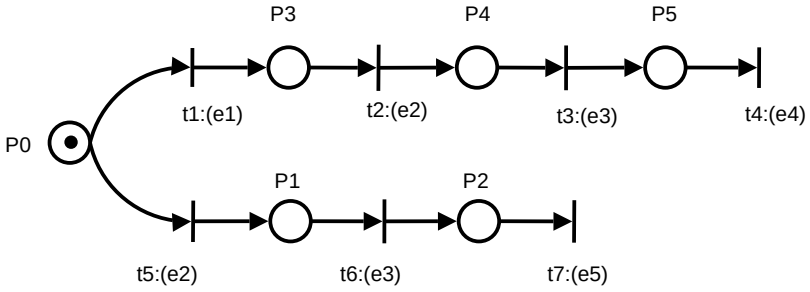


Figure 5.4 — Exemple de trajectoires possibles

La longueur de la séquence d'événements w_i est définie comme étant le nombre d'événements à conserver à partir d'un marquage certain M_i pour confirmer que le modèle suit bien la bonne trajectoire (dans l'exemple $w_i = w_0 = 4$).

Dans un premier temps, le calcul de la longueur de la séquence lorsqu'il existe qu'une seule trajectoire est présenté puis le cas où il existe deux trajectoires à partir d'un même marquage sera étudié et généralisé.

5.1.1.1 Cas d'une seule trajectoire

Si, à partir d'un marquage certain M_i, une seule trajectoire est possible alors seul la "bonne" trajectoire peut être suivie. Donc, soit l'événement reçu e_i appartient à l'ensemble des événements attendus i.e. $e_i \in E_{exp}(M_i)$ et il est dit confirmé, soit l'événement n'appartient pas à cet ensemble et il y détection. S'il est confirmé alors le marquage suivant M_{i+1} est

considéré comme certain. Un seul événement est conservé, l'événement reçu. La longueur w_i de la séquence vaut donc 1.

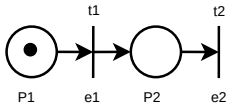


Figure 5.5 — Exemple d'une seule trajectoire

Par exemple figure 5.5, le marquage certain est $M_1 = \begin{pmatrix}1 & 0\end{pmatrix}^T$. La réception de e_1 fait passer le modèle dans le marquage $M_2 = \begin{pmatrix}0 & 1\end{pmatrix}^T$. La réception de l'événement suivant i.e. e_2 confirme que l'événement e_1 ne s'est pas inséré. La marquage M_2 est donc considéré comme certain. Si l'événement $e_i \notin E_{exp}(M_1)$ (i.e. $e_i \neq e_1$), alors il y a détection d'une incohérence. Le nombre d'événements qui doit être conservé à partir du marquage certain M_1 vaut donc $w_1 = 1$.

5.1.1.2 Cas de plusieurs trajectoires à partir d'un même marquage

Dans un premier temps, le cas de deux trajectoires possibles est d'abord étudié. Par la suite, le mécanisme à plus de deux trajectoires sera généralisé. Soit M_i, M_{p1}, M_{p2} trois marquages du modèle et s_1, s_2 deux séquences possibles de franchissement telles que :

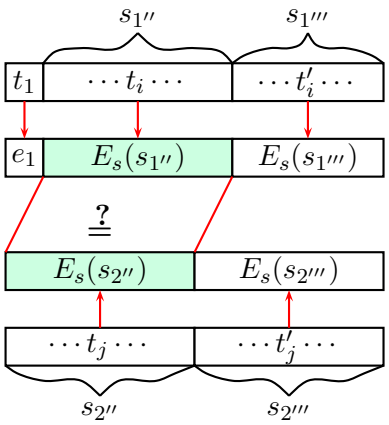
$$M_i \overset{s_1}{\longmapsto} M_{p1}$$
$$M_i \overset{s_2}{\longmapsto} M_{p2}$$

Une séquence de franchissements est définie à partir d'un marquage donné. C'est une suite de transitions franchissables successivement (sans autre franchissement de transitions). Par exemple, supposons t_1 franchissable à partir de M_0 et qui définit le nouveau marquage M_1 tel que $M_0 \overset{t_1}{\rightarrow} M_1$. De même, supposons t_2 franchissable à partir de M_1 et qui définit le nouveau marquage M_2 tel que $M_1 \overset{t_2}{\rightarrow} M_2$. Alors la séquence $s = t_1 t_2$ est une séquence de franchissement à partir du marquage M_0 ($M_0 \overset{t_1 t_2}{\rightarrow} M_2$).
Les séquences s_1 et s_2 peuvent être décomposées de la manière suivante :

$$s_1 = t_1 s_{1'}$$
$$s_2 = t_2 s_{2'}$$

Le but est de déterminer si, en supprimant le premier élément d'une séquence, par exemple de s_1, il existe une sous-séquence $s_{1''}$ de s_1 et une sous-séquence $s_{2''}$ de s_2 de longueurs égales ($longueur(s_{1''})$=$longueur(s_{2''})$) dont les événements associés sont identiques. En d'autres termes, nous voulons savoir si il existe une séquence s_1 et une séquence s_2, à partir du marquage certain M_i, telles que :

$$\begin{array}{l} s_1 = t_1 s_{1''} s_{1'''} \\ s_2 = s_{2''} s_{2'''} \end{array} \quad avec \quad \begin{cases} E_s(s_{1''})[q] = E_s(s_{2''})[q], \forall q \in [1...longueur(s_{1''})] \\ E_s(s_{1'''})[1] \neq E_s(s_{2'''})[1] \end{cases}$$



: s_1 (séquence de transitions)

: $E_s(s_1)$ (séquence d'événements)

: $E_s(s_2)$ (séquence d'événements)

: s_2 (séquence de transitions)

Si l'événement associé à la transition t1 est supposé absent, la séquence $E_s(s_1)$ privée de son premier élément ne peut être différenciée de la séquence $E_s(s_2)$ que lorsque l'événement $E_s(s_{1'''})[1]$ sera reçu.
Dès la réception de cet événement, une incohérence sera détectée.

C'est cette séquence $s_{1''}$ qui détermine le nombre d'événements à conserver. D'où :

$$w_i = longueur(E_s(t_1 s_{1''}) + E_s(s_{1'''})[1]) = longueur(E_s(t_1 s_{1''})) + 1 = longueur(E_s(s_{1''})) + 2$$

L'événement qui permet de faire la distinction entre les deux séquences i.e. ici, $E_s(s_{1'''})[1]$ est pris en compte dans la séquence.

La même procédure est appliquée en supprimant le premier élément de l'autre séquence (s_2). Comme nous voulons être sûr d'avoir la séquence d'événements dans laquelle s'est produite l'insertion ou l'absence, seul le maximum entre les deux valeurs obtenues sera considéré.

Exemple Reprenons l'exemple figure 5.4. Le marquage $M_0 = \begin{pmatrix}1 & 0 & 0 & 0 & 0\end{pmatrix}^T$ est supposé certain. A partir de M_0, deux séquences de transitions sont possibles (avec les deux séquences d'événements associées) :

$$s_1 = t_1 t_2 t_3 t_4 \text{ et } s_2 = t_5 t_6 t_7$$
$$E_s(s_1) = e_1 e_2 e_3 e_4 \text{ et } E_s(s_2) = e_2 e_3 e_5$$

s_1 et s_2 peuvent être mises sous la forme définie précédemment c'est-à-dire :

$$s_1 = t_1 s_{1''} s_{1'''} \text{ avec } s_{1''} = t_2 t_3 \text{ et } s_{1'''} = t_4$$
$$s_2 = s_{2''} s_{2'''} \text{ avec } s_{2''} = t_5 t_6 \text{ et } s_{2'''} = t_7$$

avec

$$E_s(s_{1''}) = E_s(s_{2''}) = e_2 e_3 \text{ et } E_s(s_{1'''})[1] = e_4 \neq E_s(s_{2'''})[1] = e_5$$

La longueur de la séquence à conserver à partir du marquage M_0 dans cette configuration vaut donc :

$$w_{01} = longueur(E_s(s_{1''})) + 2 = 4$$

Mettons maintenant s_1 et s_2 sous la forme :

$$s_1 = s_{1''}s_{1'''} \text{ avec } s_{1''} = \epsilon \text{ et } s_{1'''} = t_1t_2t_3t_4$$
$$s_2 = t_5s_{2''}s_{2'''} \text{ avec } s_{2''} = \epsilon \text{ et } s_{2'''} = t_6t_7$$

avec

$$E_s(s_{1''}) = E_s(s_{2''}) = \epsilon \text{ et } E_s(s_{1'''})[1] = e_1 \neq E_s(s_{2'''})[1] = e_3$$

Ainsi la longueur de la séquence à conserver à partir du marquage M_0 dans cette configuration vaut donc :

$$w_{02} = longueur(E_s(s_{1''})) + 2 = 0 + 2 = 2$$

Ainsi la longueur de la séquence à conserver à partir du marquage M_0 vaut donc $w_0 = max(w_{01}, w_{02}) = 4$. Il faut donc conserver quatre événements pour pouvoir être sûr que la bonne trajectoire est suivie et que le marquage suivant est certain.
La méthode proposée prend en compte l'ensemble des défaillances anticipées à savoir l'absence ou l'insertion d'un événement.
En effet, reprenons l'exemple 5.4. Supposons que le modèle soit dans le marquage M_0.

- Si l'événement e_1 est absent, trois événements (e_2, e_3, e_4) seront reçus avant de détecter la défaillance.
- Si l'événement e_1 est inséré, quatre événements (e_1, e_2, e_3, e_5) seront reçus avant de détecter la défaillance.
- Si l'événement e_2 est absent, il suffira d'attendre un seul événement (e_3) pour détecter la défaillance.
- Si l'événement e_2 est inséré, il suffira d'attendre deux événements (e_2, e_1) pour détecter la défaillance.

Si nous voulons être sûr que l'ensemble des cas est pris en compte, il faut conserver le nombre d'événements le plus grand. Ce nombre correspond au cas où un événement s'insère.

5.1.1.3 Généralisation à plusieurs trajectoires

En ce qui concerne la généralisation à plusieurs trajectoires, les trajectoires sont comparées deux à deux comme précédemment et le maximum entre les différents w_i obtenus est conservé.

Soit M_i un marquage certain du réseau de Petri. S_k défini l'ensemble des séquences de transitions de longueurs k, $k \in \mathbb{N}$, c'est-à-dire de la forme $s_k = t_1t_2...t_k$.
Notons que S_1 est l'ensemble des séquences de transitions de longueur 1 c'est-à-dire l'ensemble des transitions t_i sensibilisées à partir du marquage certain M_i.

Remarquons que nous parlons de marquages et non de places. Ainsi, notre raisonnement se base, non pas sur le réseau de Petri, mais sur le graphe des marquages du modèle.

Un algorithme (algo. 5.1) est donné. Cette méthode effectue $n \times (n-1)$ opérations. Ces algorithmes auront donc une complexité polynômiale en n^2.

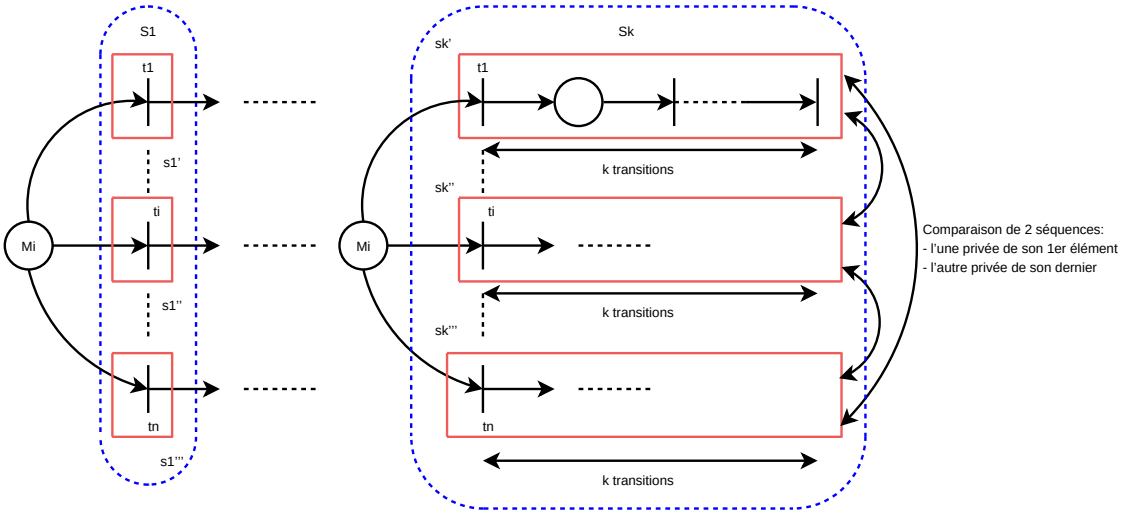


Figure 5.6 — Séquences de transitions

Exemple Prenons l'exemple de la figure 5.7. Dans ce modèle, l'événement associé à la transition t_5 a été modifié et remplacé par (e_3) par rapport à l'exemple du chapitre précédent.

À partir des marquages $M_1 = \begin{pmatrix}1 & 0 & 0 & 0 & 0\end{pmatrix}^T$, $M_3 = \begin{pmatrix}0 & 0 & 1 & 0 & 0\end{pmatrix}^T$, $M_4 = \begin{pmatrix}0 & 0 & 0 & 1 & 0\end{pmatrix}^T$, et $M_5 = \begin{pmatrix}0 & 0 & 0 & 0 & 1\end{pmatrix}^T$, il n'existe qu'une seule trajectoire possible. Par conséquent, à partir de chacun de ces marquages, un seul événement est nécessaire pour confirmer que l'événement reçu est bien issu du bon comportement ($w_1 = w_3 = w_4 = w_5 = 1$). Seul le marquage $M_2 = \begin{pmatrix}0 & 1 & 0 & 0 & 0\end{pmatrix}^T$ sensibilise deux transitions t_2 et t_4.

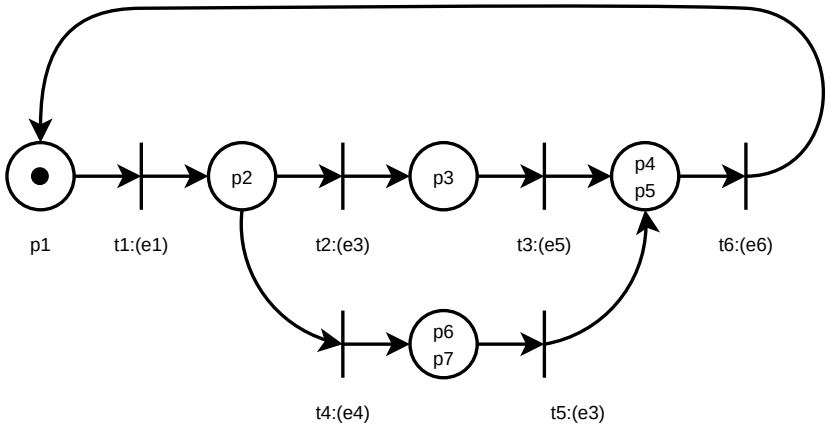


Figure 5.7 — Exemple de réseau de Petri

Soit M_2 le marquage certain. Soit $S_1 = \{t_2, t_4\}$ l'ensemble des transitions sensibilisées à partir de M_2. Nous avons également k=1 et $S = \emptyset$. Appliquons l'algorithme 5.1.

7. Considérons t_2.
8. L'événement associé est e_3.
9. Il existe deux séquences de transitions dans S_1 : les séquences t_2 et t_4 de longueur un.
11. Les marquages atteints à partir de M_2 et des transitions t_2 et t_4 sont M_3 et M_5

Algorithme 5.1 — Méthode pour le calcul de la longueur de la séquence d'événements à conserver à partir d'un marquage M_i

Entrées : Le marquage M_i, la matrice d'incidence C, l'ensemble des transitions T et l'ensemble des événements attendus au marquage M_i i.e. $E_{TFT}(M_i)$.

Sorties : Le couple (M_i, w_i)

```
1  M_i est un marquage certain.
2  S_1 est l'ensemble des transitions sensibilisées à partir de M_i.
3  k = 1 et S = ∅.
4  si card(S_1) = 1 alors
5  |  w_i vaut 1
6  sinon
7  |  pour chaque transition t_j de S_1 faire
8  |  |  Déterminer l'événement e_i associé.
9  |  |  tant que S_k ≠ ∅ faire
10 |  |  |  pour chaque séquence s_k de S_k faire
11 |  |  |  |  Déterminer le marquage M_{i+1} atteint à partir de M_i et s_k.
12 |  |  |  |  Construire les séquences s_{k+1} tel que s_{k+1} = s_k.t_i pour tout t_i
   |  |  |  |  sensibilisée à partir du marquage M_{i+1}.
13 |  |  |  fin
14 |  |  |  k=k+1.
15 |  |  |  Définir l'ensemble S_k contenant tous les s_{k+1} construites.
16 |  |  |  Définir l'ensemble S_k^1 contenant tous les s_{k+1} dont le premier élément
   |  |  |  est tel que E_t(s_{k+1})[1] = e_i.
17 |  |  |  Définir l'ensemble S_k^2 le complémentaire de S_k^1 dans S_k.
18 |  |  |  Comparer deux à deux les éléments s_{k+1}^1 de S_k^1 privées de leur premier
   |  |  |  élément et les éléments s_{k+1}^2 de S_k^2 privées de leur dernier élément.
19 |  |  |  si les deux séquences sont égales alors
20 |  |  |  |  Mettre s_{k+1}^1 et s_{k+1}^2 dans S
21 |  |  |  fin
22 |  |  |  S_k = {s, s ∈ S}.
23 |  |  |  S = ∅.
24 |  |  fin
25 |  |  Conserver le k obtenu.
26 |  fin
27 |  w_i vaut le maximum des k obtenus.
28 fin
```

respectivement.

12. t_3 est la seule transition sensibilisée à partir de M_3 et t_5 la seule à partir de M_5. Deux nouvelles séquences de longueur deux sont obtenues : $t_2.t_3$ et $t_4.t_5$.

14. k=2.

15. Le résultat est $S_2 = \{t_2.t_3, t_4.t_5\}$

16. D'où $S_2^1 = \{t_2.t_3\}$

17. et $S_2^2 = \{t_4.t_5\}$

18. Il y a deux séquences d'événements. Si la séquence de S_2^1 privée de son premier élément est comparée à celle de S_2^2 privée de son dernier élément, alors $E_s(t_3) = E_{TFE}(t_3) = e_5 \neq E_s(t_4) = E_{TFE}(t_4) = e_4$.
22. S_2 devient donc un ensemble vide.
23. $S = \emptyset$.
25. Comme S_k est vide, il en résulte k=2.

7. Considérons à présent la transition t_4.
8. L'événement associé est e_4.
9. Il existe deux séquences de transitions dans S_1 : les séquences t_2 et t_4 de longueur un.
11. Les marquages atteints à partir de M_2 et des transitions t_2 et t_4 sont M_3 et M_5 respectivement.
12. t_3 est la seule transition sensibilisée à partir de M_3 et t_5 la seule à partir de M_5. Deux nouvelles séquences de longueur deux sont à nouveau obtenues : $t_2.t_3$ et $t_4.t_5$.
14. k=2.
15. Le résultat est $S_2 = \{t_2.t_3, t_4.t_5\}$
16. D'où $S_2^1 = \{t_4.t_5\}$
17. et $S_2^2 = \{t_2.t_3\}$
18. Deux séquences d'événements sont obtenues. Si la séquence de S_2^1 privée de son premier élément est comparée à celle de S_2^2 privée de son dernier élément, nous obtenons que $E_s(t_5) = E_{TFE}(t_5) = e_3 = E_s(t_2) = E_{TFE}(t_2) = e_3$.
20. Les deux séquences sont conservées dans S.
22. $S_2 = S$.
23. $S = \emptyset$.
9. Il existe deux séquences de transitions dans S_2 : les séquences $t_2.t_3$ et $t_4.t_5$ de longueur deux.
11. Le marquage atteint à partir de M_2 et des séquences de transitions $t_2.t_3$ et $t_4.t_5$ est M_4.
12. t_6 est la seule transition sensibilisée à partir de M_4. Deux nouvelles séquences de longueur trois sont obtenues : $t_2.t_3.t_6$ et $t_4.t_5.t_6$.
14. k=3.
15. Nous obtenons $S_3 = \{t_2.t_3.t_6, t_4.t_5.t_6\}$.
16. D'où $S_3^1 = \{t_4.t_5.t_6\}$
17. et $S_3^2 = \{t_2.t_3.t_6\}$.
18. Deux séquences d'événements sont obtenues. Si la séquence de S_3^1 privée de son premier élément est comparée à celle de S_3^2 privée de son dernier élément, nous obtenons que $E_s(t_5.t_6) = E_{TFE}(t_5).E_{TFE}(t_6) = e_3.e_6 \neq E_s(t_2.t_3) = E_{TFE}(t_2).E_{TFE}(t_3) = e_3.e5$.
22. $S_3 = S = \emptyset$.
23. $S = \emptyset$.
25. Comme S_k est vide, le résultat obtenu est k=2.
27. Au final, le résultat est $w_2 = max(2, 3) = 3$.

Une méthode pour le calcul de la longueur de la séquence d'événements à conserver à partir d'un marquage a été présentée. Ceci est fait en hors-ligne pour tous les marquages. Cette longueur dépend du marquage où le modèle se trouve. Lors de la surveillance, le modèle évolue de marquage en marquage. La longueur de la séquence à conserver dépend donc également

des marquages précédents. Il faut donc, lors de la phase en ligne, calculer la longueur de la séquence d'événements à conserver en fonction du marquage courant c'est-à-dire M_i et donc w_i, mais également des marquages précédents.

5.1.2 Détermination de la longueur à conserver lors de la surveillance en ligne

L'objectif dans le calcul de la longueur de la séquence est d'avoir un minimum d'événements à examiner lors du mécanisme de localisation. À partir d'un marquage certain M_i, w_i événements doivent être attendus pour confirmer que l'événement reçu en M_i est issu du bon fonctionnement. Une fois confirmé, cet événement peut être supprimé de la séquence d'événements conservés.

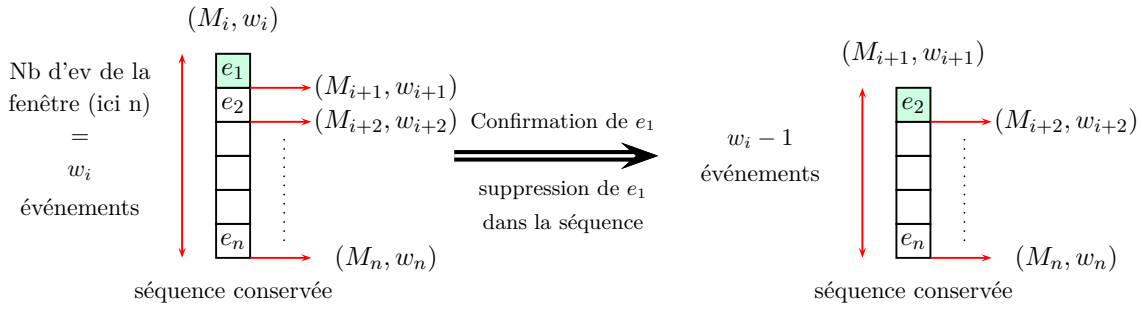


Figure 5.8 — Mise à jour de la longueur de la séquence

Le nombre d'événements nécessaires (soit w_{i+1} associé au marquage suivant M_{i+1} conservé) est considéré pour confirmer l'événement suivant (ici e_2). Deux possibilités existent.

- Soit le nombre d'événements nécessaires w_{i+1} est plus petit ou égal au nombre d'événements déjà conservés $w_i - 1$ et dans ce cas le premier événement peut être supprimé de la séquence conservée. La procédure est réitérée pour l'événement suivant.

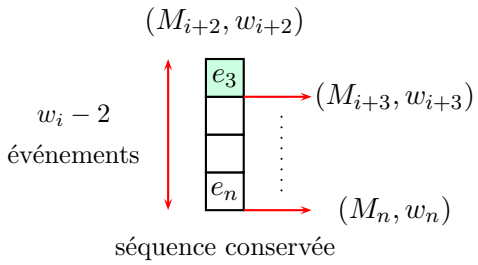


Figure 5.9 — Cas où $w_i - 1 \geq w_{i+1}$

- Soit le nombre d'événements w_{i+1} nécessaires est plus grand que le nombre d'événements conservés et donc il faut attendre que le nombre d'événements conservés soit égal à w_{i+1} (les prochains événements reçus doivent être ajoutés dans la séquence conservée).

Par conséquent, la longueur de la séquence conservée varie pour ne garder finalement que les événements qui ne sont pas encore confirmés, et ce toujours dans l'hypothèse de l'insertion

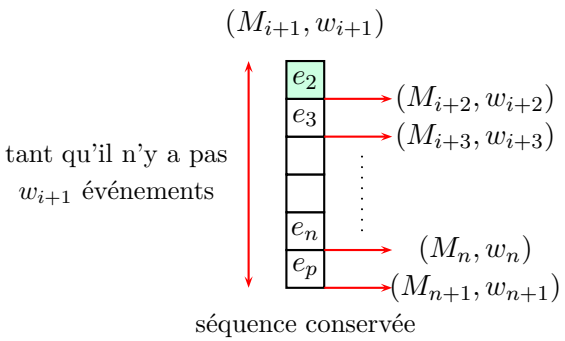


Figure 5.10 — Cas où $w_i - 1 < w_{i+1}$

ou l'absence d'un seul événement.

Reprenons les différentes étapes. Soit un marquage certain M_i. À ce marquage est associé un nombre w_i qui correspond au nombre d'événements à attendre avant de pouvoir affirmer que l'événement reçu à ce marquage n'est pas issu d'une défaillance. Il est conservé autant d'événements que nécessaire ainsi que les différents marquages (et les w_i associés) atteints. Une fois que le nombre d'événements est égal à w_i, alors nous pouvons affirmer que l'événement reçu au marquage M_i n'est pas issu d'une défaillance. Il ne peut donc être supprimé de la séquence d'événements conservée. Avant la réception d'un autre événement, le marquage qui suivait M_i est considéré. Si le nombre d'événements nécessaires pour confirmer l'événement reçu à ce marquage est inférieur au nombre d'événements dans la séquence alors nous pouvons affirmer que cet événement n'est pas issu d'une défaillance. Si c'est le cas, alors il peut être supprimé de la séquence conservée. Puis, toujours avant la réception d'un nouvel événement, le marquage suivant est à nouveau considéré et le processus est réitéré. Si ce n'est pas le cas alors de nouveaux événements sont attendus jusqu'à ce que le nombre d'événements dans la séquence conservée soit égal à celui attendu.

Un algorithme (algo. 5.2) est donné. Soit S_{cons} la séquence des événements conservés. Soit w_c la longueur de la séquence conservée, M_i le marquage courant, M_0 le marquage initial supposé connu.

Exemple Reprenons l'exemple de la figure 5.7. Les longueurs w_i déterminées sont :

$$\begin{array}{ll} M_1 \longmapsto w_1 = 1 & M_4 \longmapsto w_4 = 1 \\ M_2 \longmapsto w_2 = 3 & M_5 \longmapsto w_5 = 3 \\ M_3 \longmapsto w_3 = 1 & \end{array}$$

Supposons que le modèle se trouve dans le marquage initial $M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix}^T$ et que e_1 soit reçu.

Séquence conservée		$Marquage$	w
$S_{cons} =$	$[]$	M_1	1
$S_{cons} =$	$[e_1]$	M_2	3

Algorithme 5.2 — Calcul en ligne de la longueur de la séquence d'événements à conserver

Entrées : La matrice d'incidence C et la matrice d'incidence avant I, le marquage initial (certain) M_0, les couples (M_i, w_i) et les événements e_i reçus.

Sorties : l'ensemble des événements conservés S_{cons}, l'ensemble des marquages conservés M.

```
1  w_c = 0, M_i = M_0, M = [M_i], w_i = w_0, w = [w_i], S_cons = [ ],
2  tant que il n'y a pas de détection faire
3      si réception d'un événement e_i alors
4          S_cons = [S_cons; e_i]
5          si e_i ∉ E_exp(M_i) alors
6              Exit ; % Incohérence détectée
7          sinon
8              M_{i+1} = M_i + C(., E_F(M_i, e_i))
9              w = [w; w_{i+1}]
10             M = [M; M_{i+1}]
11             w_c = w_c + 1
12             tant que w_c > w[1] faire
13                 S_cons = S_cons[2...end]
14                 w = w[2...end]
15                 M = M[2...end]
16                 w_c = w_c − 1
17             fin
18             M_i = M_{i+1}
19         fin
20     fin
21 fin
```

Supposons que e_4 soit reçu quand le modèle se trouve dans le marquage M_2. Le nombre d'événements w_c de la séquence est égal au nombre d'événement à attendre à savoir $w_c = w_1$. Le premier événement de la séquence S_{cons} peut donc être supprimé. w_2 événements doit être attendus.

Séquence conservée		$Marquage$	w
$S_{cons} =$	$[]$	M_1	$w_1 = 1$
$S_{cons} =$	$[e_1]$	M_2	$w_2 = 3$
$S_{cons} =$	$[e_4]$	M_5	$w_5 = 1$
$S_{cons} =$	$[e_4, e_3]$	M_4	$w_4 = 1$
$S_{cons} =$	$[e_4, e_3, e_6]$	M_1	$w_2 = 1$

Lorsque le modèle se trouve dans le marquage M_1, la séquence conservée est $S_{cons} = [e_4, e_3, e_6]$. À partir de là, nous avons atteint $w_c = w_2$. Le premier événement de la séquence soit e_4 peut être supprimé. Il reste donc deux événements dans la séquence. Le marquage suivant étant M_5 et le w_5 associé valant 1, alors nous sommes dans le cas où $w_c \geq w_5$. Par conséquent, le premier élément de S_{cons} i.e. e_3 peut être supprimé. Finalement, il ne reste plus que $S_{cons} = [e_6]$. Un seul événement se trouve dans la séquence. L'algorithme s'arrête dès qu'il y a une détection, et est réinitialisé ainsi que le marquage.

Dans le chapitre précédent, nous avons exposé le principe de détection. Dans ce chapitre, nous voulons présenter le principe de localisation. Dans un premier temps, une technique qui consiste à conserver un certain nombre d'événements tant qu'il n'a pas été confirmé qu'ils ne sont pas issus d'une défaillance a été établie. Pour pouvoir l'affirmer, il faut recevoir un certain nombre d'événements sans avoir eu de détection. Cette séquence évolue donc en fonction du marquage courant et des marquages par lesquels le modèle est passé. Cette séquence de marquages M_i (en leur associant le nombre d'événements w_i à conserver) est conservée. Lorsqu'il y a une défaillance qui se traduit par l'insertion ou l'absence d'un seul événement, alors cette insertion ou cette absence n'a pu avoir lieu que dans la séquence d'événements conservée. Il faut donc à présent déterminer cet événement "fautif".

5.2 Principe de localisation

Par hypothèse, un événement peut s'inséré ou être absent. Cet événement se situe dans la séquence conservée. L'objectif de cette étape de localisation est de déterminer cet événement. Pour cela, il faut rétablir la cohérence entre les trajectoires du modèle et la séquence de transitions associée à la séquence d'événements reçus, en supprimant ou en insérant un événement dans cette séquence. Un test de cohérence est donc fait sur toute séquence obtenue en supprimant ou en insérant un événement. Un ensemble d'hypothèses expliquant les observations reçues est alors obtenu.

5.2.1 Insertion d'un événement

Au fur et à mesure de l'évolution du modèle, une séquence d'événements est conservée. Elle est notée S_{cons}. La séquence des couples (M_i,w_i), i.e. les marquages par lesquels le modèle est passé et le nombre d'événements à conserver w_i associé sont également conservés (figure 5.11).

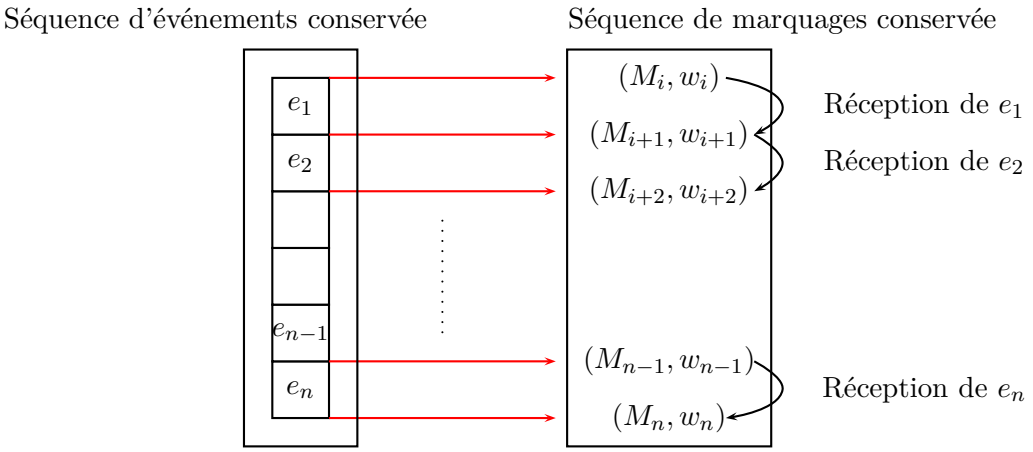


Figure 5.11 — Séquences d'événements et de marquages conservées

Si un événement s'est inséré, alors sa suppression permet de rétablir la cohérence avec une trajectoire du modèle. Supprimons un par un les éléments de la séquence conservée (figure 5.12).

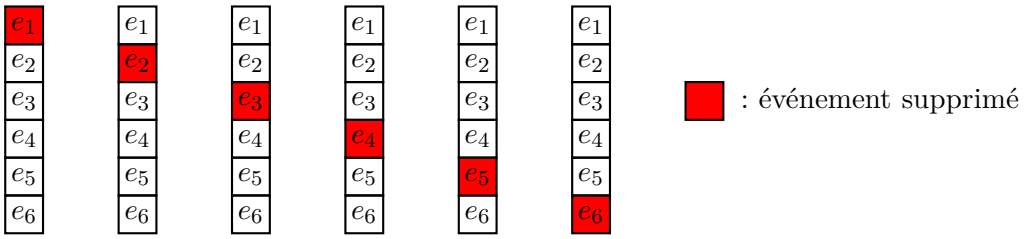


Figure 5.12 — Test sur toute séquence obtenue en supprimant un événement

Pour chaque événement supprimé, il faut vérifier que la nouvelle séquence d'événements obtenue est associée à une séquence de franchissement du modèle.
Étant donné que les événements de la séquence S_{cons} sont supprimés successivement, il n'est pas nécessaire de vérifier que les événements précédant celui qui a été supprimé soient associés à des transitions franchissables. Le marquage dans lequel le modèle se trouvait lorsque l'événement a été reçu est alors considéré. Il faut vérifier que la nouvelle séquence (celle qui commence à partir de l'événement qui suit celui qui a été supprimé) est associée à une séquence de franchissement à partir de ce marquage (fig : 5.13).
L'algorithme 5.3 a été développé.

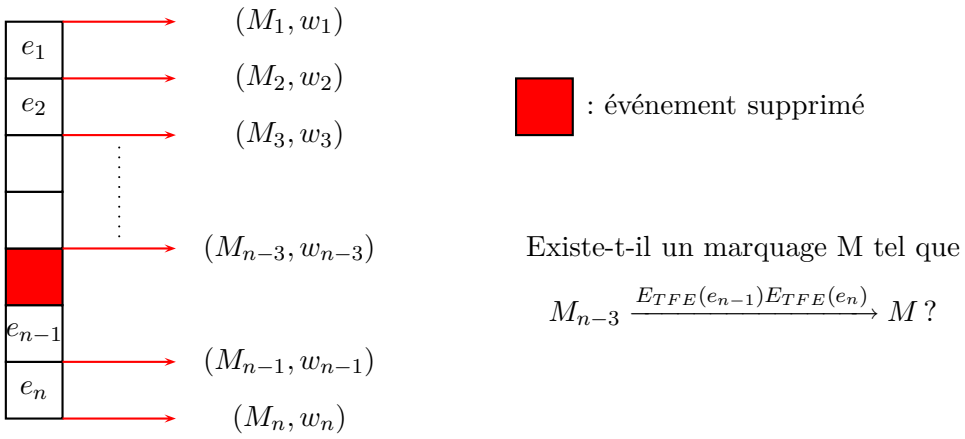


Figure 5.13 — Principe de rétablissement de la cohérence par suppression d'un événement

Algorithme 5.3 — Algorithme de localisation de l'événement par rétablissement de cohérence pour l'insertion d'un événement

Entrées :

l'ensemble des événements conservés S_{cons},

l'ensemble des marquages conservés M_{cons},

n le cardinal de S_{cons}

Sorties : l'ensemble des événements suspectés $E_{suspects}$

```
1  pour chaque élément e_k de S_cons faire
2      Considérer la séquence S = S_cons[k, ..., n-1] = [e_{k+1}, ..., e_n],
3      Considérer le marquage M = M_cons[k-1],
4      pour chaque élément e de S faire
5          si E_TFT(M) ∩ E_TFE(e) ≠ ∅ alors
6              Définir le nouveau marquage M tel que :
7              M = M + C(., E_TFE(e))
8          sinon
9              Exit
10         fin
11     fin
12     si tous les éléments de S ont été traités alors
13         E_suspects = E_suspects ∪ {e_k}
14     fin
15 fin
```

Cette méthode permet de déterminer un ensemble d'événements suspects qui se seraient insérés. Leur suppression dans la séquence d'événements reçus rétablit la cohérence avec une trajectoire du modèle. À présent, il faut déterminer l'ensemble des événements qui sont susceptibles d'être absents et qui, par leur présence, rétabliraient la cohérence avec les trajectoires du modèle.

5.2.2 Absence d'un événement

Dans le cas où un événement est supposé absent, l'ajout de celui-ci dans la séquence d'événements permettra de rétablir la cohérence avec une trajectoire du modèle. Insérons un à un les événements du modèle dans la séquence d'événements reçus S_{cons} (fig. 5.14).

Pour chaque marquage conservé, les événements **attendus** par ce marquage sont insérés successivement. Il faut ensuite vérifier que la nouvelle séquence obtenue est associée à une séquence de franchissement du modèle. Étant donné qu'un événement est ajouté successivement à la séquence S_{cons}, il n'est pas nécessaire de vérifier que les événements, précédant celui qui a été ajouté, soient associés à des transitions franchissables. Le marquage dans lequel le modèle se trouve lorsque l'événement est supposé absent est considéré et une

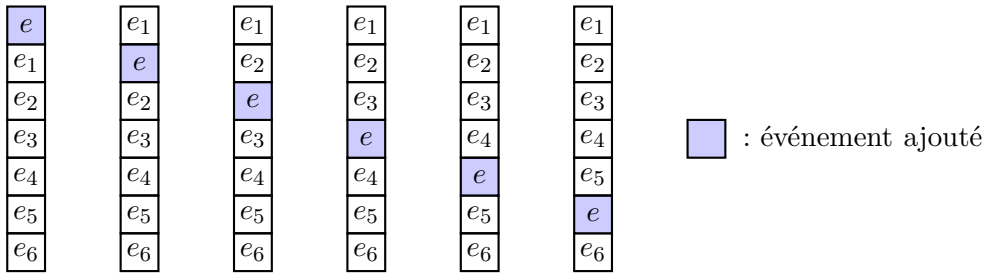


Figure 5.14 — Test sur toute séquence obtenue en ajoutant un événement

vérification est effectuée pour savoir si la nouvelle séquence (celle qui commence à partir de l'événement ajouté) est associée à une séquence de franchissement à partir de ce marquage (fig : 5.15).

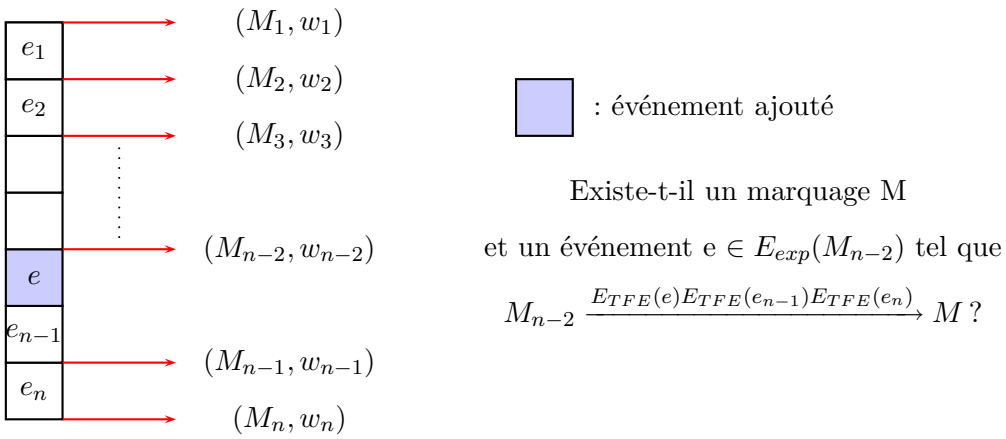


Figure 5.15 — Principe de rétablissement de la cohérence par l'ajout d'un événement

L'algorithme 5.4 a été développé.

Algorithme 5.4 — Algorithme de localisation de l'événement par rétablissement de cohérence pour l'absence d'un événement

Entrées :

l'ensemble des événements conservés S_{cons},

l'ensemble des marquages conservés M_{cons},

n le cardinal de S_{cons}

Sorties : l'ensemble des événements suspectés $E_{suspects}$

```
1  pour chaque marquage M_k de M_cons faire
2      M = M_k pour chaque événement e de E_exp(M_k) faire
3          Considérer la séquence S = [e, e_{k+1}, ..., e_n],
4          pour chaque élément e' de S faire
5              si E_TFT(M) ∩ E_TFE(e') ≠ ∅ alors
6                  Définir le nouveau marquage M tel que :
7                  M = M + C(., E_TFE(e'))
8              sinon
9                  Exit
10             fin
11         fin
12         si tous les éléments de S ont été traités alors
13             E_suspects = E_suspects ∪ {e_k}
14         fin
15     fin
16 fin
```

Cette méthode permet de déterminer un ensemble d'événements suspects qui seraient absents. Leur ajout dans la séquence d'événements reçus rétablit la cohérence avec une trajectoire du modèle.

5.3 Conclusion

Nous avons présenté dans ce chapitre une méthode qui permet de localiser un événement qui a pu s'insérer dans la séquence des événements reçus ou être absent de cette même séquence.
Pour cela, il faut déterminer, dans un premier temps en hors-ligne, le nombre d'événements à conserver à partir d'un marquage pour confirmer que l'événement reçu à ce marquage correspond bien à un événement issu du bon comportement.
Ensuite, en-ligne, lors de la surveillance du système, le nombre d'événements qu'il faut effectivement conserver est calculé de façon dynamique, car ce nombre dépend des états par lesquels le modèle est passé.
Ainsi, lorsqu'une détection est faite, une séquence d'événements dans laquelle un événement a

pu s'insérer ou bien être absent, et qui pourrait expliquer l'incohérence observée, est conservée. Une méthode exécutée en ligne a donc été proposée permettant de déterminer ces événements insérés ou absents.
L'inconvénient de cette méthode est que la plupart des calculs se font en ligne, ce qui pour une application embarquée n'est pas l'approche la plus adaptée.
En outre, la méthode proposée ne fait aucune hypothèse quant à la convergence du calcul du nombre d'événements nécessaires w_i à partir d'un marquage M_i. En effet, par exemple sur la figure 5.16, il apparaît que w_1 ne peut être calculé par notre méthode. En effet, le calcul rentre dans une boucle infinie. Ceci est dû au fait que l'insertion de e_1 ou son absence ne peut pas être détectée. Cela implique qu'il faut faire une étude de détectabilité/diagnosticabilité a priori sur modèle.
C'est pourquoi, dans le chapitre suivant, nous proposons une approche qui permette de faire moins de calculs en ligne en suivant simplement le comportement d'un modèle. Elle permet également de s'affranchir de ces problèmes de détectabilité/diagnosticabilité d'un point de vu convergence.

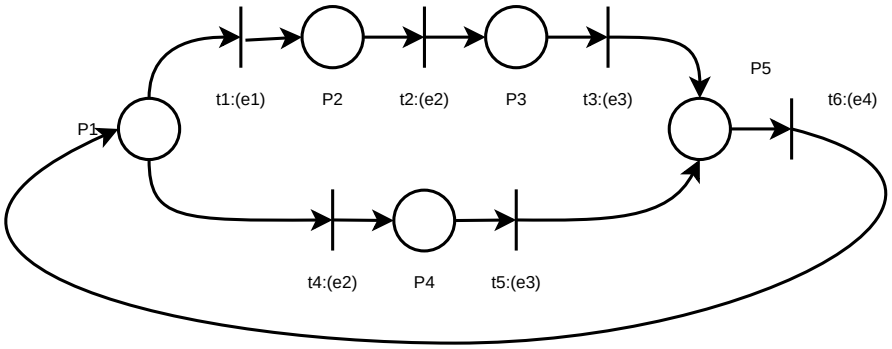


Figure 5.16 — Problème de boucle

6 Approche diagnostiqueur : les automates

Introduction

Dans le chapitre précédent, une technique a été proposée permettant de déterminer un événement absent ou inséré à partir d'une fenêtre d'événements qui évolue lors de la surveillance en ligne du système. Cette méthode suppose de nombreux calculs en ligne, ce qui pour une application embarquée est préjudiciable. Pour éviter ce genre de calculs, une approche basée sur l'approche diagnostiqueur (cf. chapitre 3.2) est présentée dans ce chapitre. Cette approche compile les informations de diagnostic dans un modèle. L'état de défaillance dans lequel le système se trouve est déterminé en suivant le modèle.

Dans nos travaux, aucune connaissance sur les défaillances intermittentes est disponibles contrairement aux approches diagnostiqueur classiques. Une hypothèse consiste à supposer qu'une défaillance se traduit par l'insertion ou l'absence d'un événement observable du système. Cet événement émis traduit généralement une information concernant le comportement normal du système et non une défaillance (ce n'est pas une alarme par exemple). C'est à partir de cette hypothèse et du modèle de bon comportement que sont construits des modèles qui vont fournir des hypothèses sur les défaillances possibles pouvant expliquer les observations reçues.

La figure 6.1 présente les différentes étapes de notre approche. Dans un premier temps, un modèle incluant les fautes possibles est construit, puis des modèles de classes de fautes. Dans un troisième temps, le produit synchronisé entre ces deux modèles est calculé pour ensuite terminer par la construction du diagnostiqueur (Soldani *et al.*, 2007*a*).

Cette méthode s'apparente fortement aux travaux de (Jéron *et al.*, 2006) (cf. chapitre 3.3). Les modèles utilisés sont représentés par des automates puisque c'est le formalisme utilisé dans les approches diagnostiqueurs.

Dans un souci de modularité, les modèles sont découplés. Le fait d'adopter une démarche modulaire permet, par exemple, de compléter les modèles par des modèles de fautes si les connaissances le permettent ou encore par d'autres modèles construits sous des hypothèses différentes des nôtres (cf. figure 6.2). Ainsi, un modèle est obtenu, celui de bon comportement, qui se charge de la surveillance du système et de détecter les éventuelles incohérences. Deux

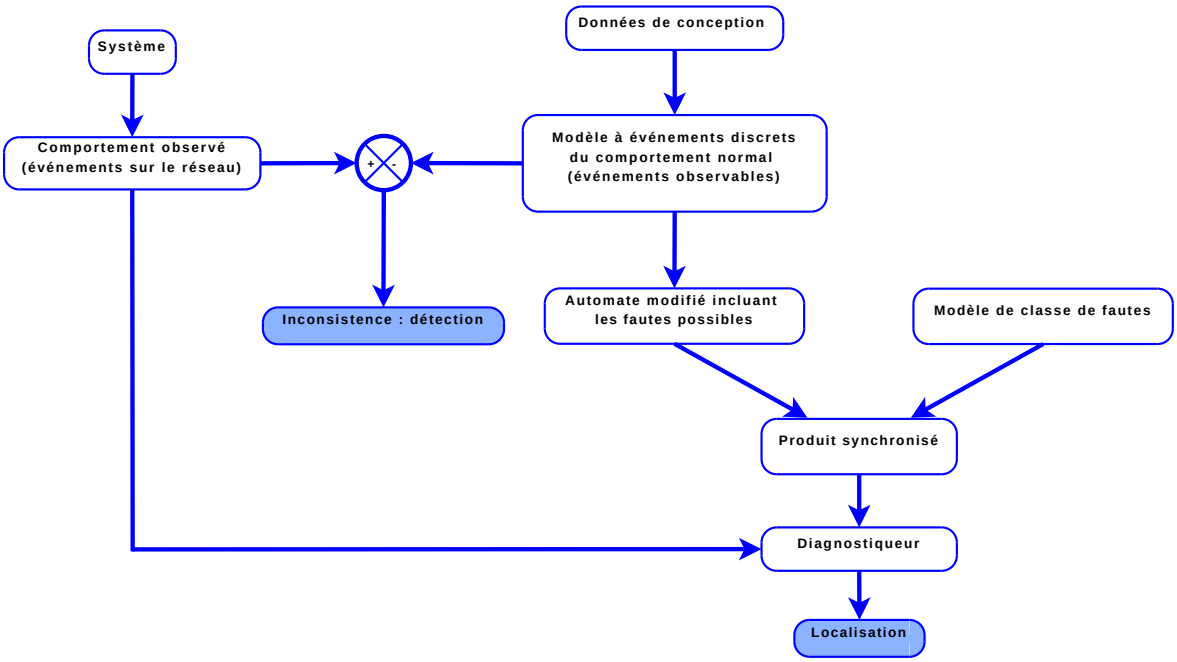


Figure 6.1 — Principe du diagnostic à base de modèles à l'aide du diagnostiqueur

diagnostiqueurs sont également construits, l'un prenant en compte l'insertion d'un événement, l'autre prenant en compte son absence. Par la suite, ces modèles peuvent être fusionnés afin de n'avoir plus qu'un seul modèle à suivre mais la construction de chacun de ces modèles se fait de façon indépendante.

Nous allons maintenant présenter les différents modèles construits à partir du modèle de bon comportement pour finalement obtenir le diagnostiqueur.

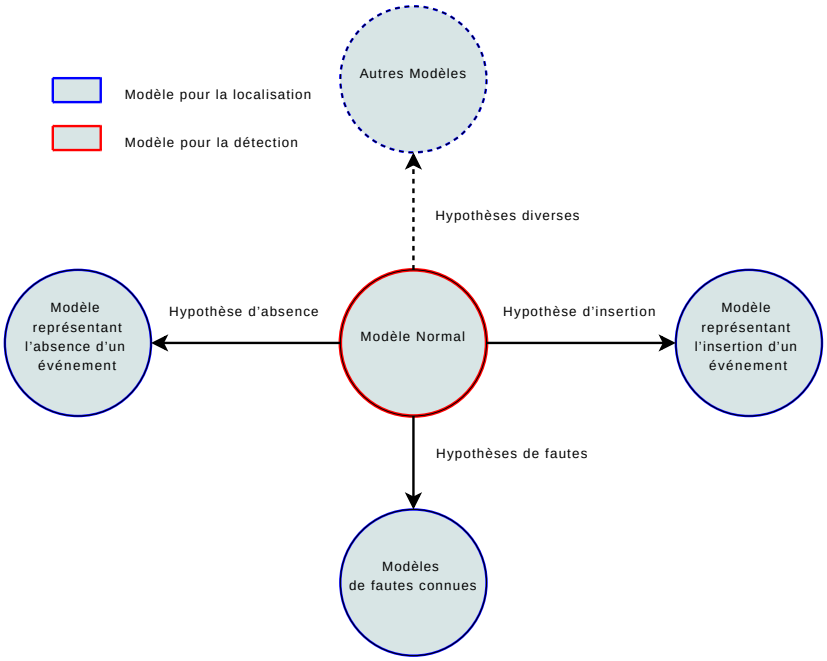


Figure 6.2 — Représentation des modèles

6.1 Modèles de Classe de Fautes

6.1.1 Le modèle

Le modèle de classe de fautes a pour objectif de représenter certaines défaillances dont les caractéristiques sont identiques. Le modèle de classe de fautes pour l'insertion d'événements prend en compte l'ensemble des insertions possibles. De même, le modèle de classe de fautes pour l'absence d'événement prend en compte l'ensemble des absences possibles. Dans ces deux modèles, l'insertion ou l'absence d'un seul événement est considérée. Le modèle d'insertion d'événements et le modèle d'absence d'événement sont construits et représentés sur le schéma (6.3).

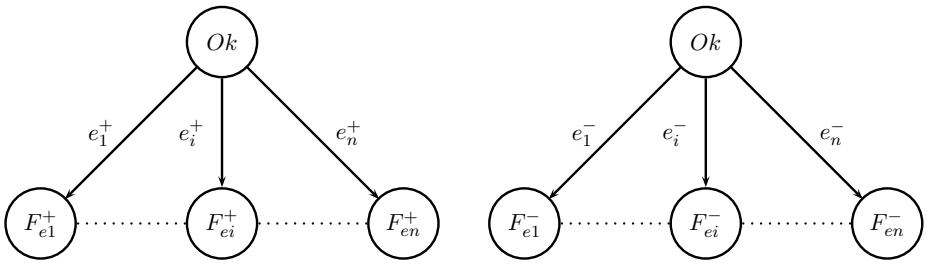


Figure 6.3 — Modèles de classe de fautes (insertion et absence respectivement)

Le modèle représente le passage d'un mode de bon comportement (Ok) vers un mode d'hypothèse de "faute" $F_{e_i}^+$ ou $F_{e_i}^-$. Ces hypothèses représentent respectivement l'insertion ou à l'absence de l'événement associé à l'arc r_i et étiqueté par e_i. Ainsi, dans un mode de bon comportement (Ok), toutes les insertions (respectivement les absences) sont supposées susceptibles de se produire. Les arcs r_i de ce modèle correspondent aux arcs r_i du modèle de bon comportement. Le modèle est donc composé d'un mode Ok et de, au plus, n_{BF}^t modes correspondant aux différentes hypothèses d'insertions (respectivement d'absences), où n_{BF}^t représente le nombre d'arcs dans le modèle de bon comportement (il peut y avoir plusieurs arcs avec le même événement). Une fois qu'une hypothèse d'insertion (ou d'absence) est faite, il n'est pas possible de revenir dans une hypothèse de bon comportement et aucune autre hypothèse n'est supposée. Cela correspond à l'hypothèse faite au début à savoir qu'un seul événement peut s'insérer ou être absent. Un modèle comportant $n_{BF}^t + 1$ états et n_{BF}^t arcs est établi. Soit n_{INS}^s et n_{INS}^t le nombre d'états et respectivement d'arcs du modèle représentant l'insertion d'un événement, et soit n_{ABS}^s et n_{ABS}^t le nombre d'états et respectivement d'arcs du modèle représentant l'absence d'un événement. Ces modèles sont notés par Γ^+ et Γ^-. Le modèle de bon comportement est noté également par Γ_{BF}.

L'automate $\Gamma^+ =< S^+, S_0^+, T^+, R^+ >$ est défini par :

- $S^+ = \{Ok, F_{e_1}^+, \ldots, F_{e_n}^+\}$,$\forall e_i \in T(\Gamma_{BF})$;
- $S_0^+ = \{Ok\}$;
- $T^+ = \left\{e_i^+, \ldots, e_n^+\right\}$, $\forall e_i \in T(\Gamma_{BF})$;

- $R^+ = \left\{(Ok, e_i^+, F_{e_i}^+)\right\} \forall r_i = (q, e_i, q') \in R(\Gamma_{BF})$.

Et pour l'automate $\Gamma^- =< S^-, S_0^-, T^-, R^- >$:
- $S^- = \{Ok, F_{e_1}^-, \ldots, F_{e_n}^-\}, \forall e_i \in T(\Gamma_{BF})$;
- $S_0^- = \{Ok\}$;
- $T^- = \left\{e_i^-, \ldots, e_n^-\right\}, \forall e_i \in T(\Gamma_{BF})$;
- $R^- = \left\{(Ok, e_i^-, F_{e_i}^-)\right\} \forall r_i = (q, e_i, q') \in R(\Gamma_{BF})$.

L'insertion d'un événement e_i associé à l'arc r_i est représentée dans Γ^+ par un arc étiqueté par e_i^+. De la même manière, l'absence d'un événement e_i associé à l'arc r_i est représentée dans Γ^- par un arc étiqueté par e_i^-. $F_{e_i}^+$ correspond à l'hypothèse que l'événement e_i associé à l'arc r_i s'est inséré. $F_{e_i}^-$ correspond à l'hypothèse que l'événement e_i associé à l'arc r_i est absent.

Remarque :
- l'événement e_i^+ est le même événement que e_i. L'événement e_i^- correspond à un événement non observable.
- les modèles de classe de fautes peuvent être construits à partir des connaissances du système. Par exemple, les modèles peuvent être modifiés en ajoutant des arcs à la suite des états F_{e_i}, ce qui correspondrait au fait qu'un événement inséré puisse être suivi d'un autre événement inséré ou alors d'un événement absent, ou autres...

6.1.2 Exemple

Prenons un petit exemple pour illustrer les différents étapes de notre approche (figure 6.4).
L'automate $\Gamma_{BF} =< S_{BF}, S_{0_{BF}}, T_{BF}, R_{BF} >$ est défini :
- $S_{BF} = \{1, 2\}$;
- $S_{0_{BF}} = \{1\}$;
- $T_{BF} = \{e_1, e_2\}$;
- $R_{BF} = \{(1, e_1, 2), (2, e_2, 1)\}$.

Soit l'automate $\Gamma^+ =< S^+, S_0^+, T^+, R^+ >$ défini à partir de Γ_{BF} :
- $S^+ = \{Ok, F_{e_1}^+, F_{e_2}^+\}, \forall e_i \in T(\Gamma_{BF}) = \{e_1, e_2\}$;
- $S_0^+ = \{Ok\}$;
- $T^+ = \left\{e_1^+, e_2^+\right\}, \forall e_i \in T(\Gamma_{BF})$;
- $R^+ = \left\{(Ok, e_1^+, F_{e_1}^+), (Ok, e_2^+, F_{e_2}^+)\right\}, \forall r \in R(\Gamma_{BF})$.

Et l'automate $\Gamma^- =< S^-, S_0^-, T^-, R^- >$ également défini à partir de Γ_{BF} :
- $S^- = \{Ok, F_{e_1}^-, F_{e_2}^-\}, \forall e_i \in T(\Gamma_{BF}) = \{e_1, e_2\}$;
- $S_0^- = \{Ok\}$;
- $T^- = \left\{e_1^-, e_2^-\right\}, \forall e_i \in T(\Gamma_{BF})$;
- $R^- = \left\{(Ok, e_1^-, F_{e_1}^-), (Ok, e_2^-, F_{e_2}^-)\right\}, \forall r \in R(\Gamma_{BF})$.

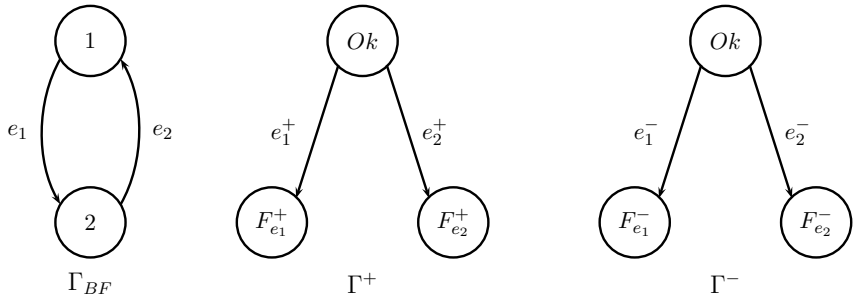


Figure 6.4 — Exemple de modèle avec ses classes de fautes

6.2 Modèles incluant des défaillances génériques

À présent, pour obtenir un modèle représentant l'évolution de ces hypothèses de défaillances (insertion ou absence), il faut modifier le modèle de bon comportement en considérant ces insertions ou ces absences.

6.2.1 Modèle pour l'absence d'événement

L'absence d'un événement est considérée comme l'occurrence d'un événement non observable. Le système peut évoluer alors que le modèle reste dans le même état.
La modification à apporter pour l'absence d'événement consiste à ajouter un nouvel arc défini par (s, e_i^-, s') à partir de tous les arcs de R_{BF} définis par (s, e_i, s'). L'événement non observable associé est e_i^-.

Le nouvel automate, noté $\Gamma_{BF}^- =< S_{BF}^-, S_{0_{BF}}^-, T_{BF}^-, R_{BF}^- >$ est défini par :
- $S_{BF}^- = S_{BF}$;
- $S_{0_{BF}}^- = S_{0_{BF}}$;
- $T_{BF}^- = T_{BF} \bigcup \left\{ e_i^- \right\}, \forall e_i \in T(\Gamma_{BF})$;
- $R_{BF}^- = R_{BF} \bigcup \left\{ (s, e_i^-, s') | \forall (s, e_i, s') \in R_{BF}, (s, e_i^-, s') \in R_{BF}^- \right\}$

6.2.2 Modèle pour l'insertion d'événement

Un événement reçu peut être fautif. Si l'événement correspond à un événement attendu alors le modèle peut évoluer alors que le système reste dans le même état.
La modification à apporter pour l'insertion d'événement consiste à ajouter un arc sur l'automate avec pour état de sortie le même que celui d'entrée. Cette modification est faite pour tout arc de R_{BF} défini par (s, e_i, s'). L'étiquette de cet arc est e_i^+ et le nouvel arc est noté (s, e_i^+, s).
L'événement fautif ne doit pas changer l'état du modèle. Un arc d'un état vers lui-même est ajouté pour tous ses arcs de sortie. En effet, les événements associés à ces arcs sont les seuls qui puissent faire changer le modèle de trajectoire sans être immédiatement détectés. À chaque occurrence d'un événement, l'hypothèse que cet événement est inséré est toujours

faite et n'est pas représentée pour les événements qui ne sont pas attendus par le modèle. Nous pourrions le prendre en compte en ajoutant à chaque état toutes les transitions du modèle mais cela complexifierait le modèle sans apporter réellement d'information.

Le nouvel automate, noté $\Gamma^+_{BF} =< S^+_{BF}, S^+_{0_{BF}}, T^+_{BF}, R^+_{BF} >$ est défini par :
- $S^+_{BF} = S_{BF}$;
- $S^+_{0_{BF}} = S_{0_{BF}}$;
- $T^+_{BF} = T_{BF} \bigcup \left\{e^+_i\right\}, \forall e_i \in T(\Gamma_{BF})$;
- $R^+_{BF} = R_{BF} \bigcup \left\{(s, e^+_i, s) | \forall (s, e_i, s') \in R_{BF}, (s, e^+_i, s) \in R^+_{BF}\right\}$.

Les deux automates ont autant d'états que l'automate de bon comportement (noté $n^s_{BF+} = n^s_{BF}$ et $n^s_{BF-} = n^s_{BF}$) et deux fois plus d'arcs (noté $n^t_{BF+} = 2 * n^t_{BF}$ et $n^t_{BF-} = 2 * n^t_{BF}$).

6.2.3 Exemple

Reprenons l'exemple figure 6.4 et construisons les deux automates modifiés (figure 6.5).

L'automate pour l'absence, noté $\Gamma^-_{BF} =< S^-_{BF}, S^-_{0_{BF}}, T^-_{BF}, R^-_{BF} >$:
- $S^-_{BF} = \{1, 2\}$
- $S^-_{0_{BF}} = \{1\}$
- $T^-_{BF} = \{e_1, e_2\} \bigcup \left\{e^-_1, e^-_2\right\} = \{e_1, e_2, e^-_1, e^-_2\}$
- $R^-_{BF} = \{(1, e_1, 2), (2, e_2, 1)\} \bigcup \left\{(1, e^-_1, 2), (2, e^-_2, 1)\right\}$
  $= \{(1, e_1, 2), (2, e_2, 1), (1, e^-_1, 2), (2, e^-_2, 1)\}$

L'automate pour l'insertion, noté $\Gamma^+_{BF} =< S^+_{BF}, S^+_{0_{BF}}, T^+_{BF}, R^+_{BF} >$:
- $S^+_{BF} = \{1, 2\}$
- $S^+_{0_{BF}} = \{1\}$
- $T^+_{BF} = \{e_1, e_2\} \bigcup \left\{e^+_1, e^+_2\right\} = \{e_1, e_2, e^+_1, e^+_2\}$
- $R^+_{BF} = \{(1, e_1, 2), (2, e_2, 1)\} \bigcup \left\{(1, e^+_1, 1), (2, e^+_2, 2)\right\}$
  $= \{(1, e_1, 2), (2, e_2, 1), (1, e^+_1, 1), (2, e^+_2, 2)\}$

6.3 Produit synchronisé des modèles

Après la construction de ces différents modèles, un produit synchronisé est effectué entre chaque modèle de bon comportement et le modèle de classes de fautes associé. Cette synchronisation associe les états de la fonction (S_1,...,S_n) aux modes (Ok, $F^+_{e_i}$) ou (Ok, $F^-_{e_i}$).

Dans un premier temps, rappelons les différentes définitions utilisées pour le produit synchronisé, qui sont largement reprises du cours de (Julliand, 2003).

6.3.1 Rappels

Tout d'abord, nous définissons l'étiquette vide ϵ telle que (s, ϵ, s).

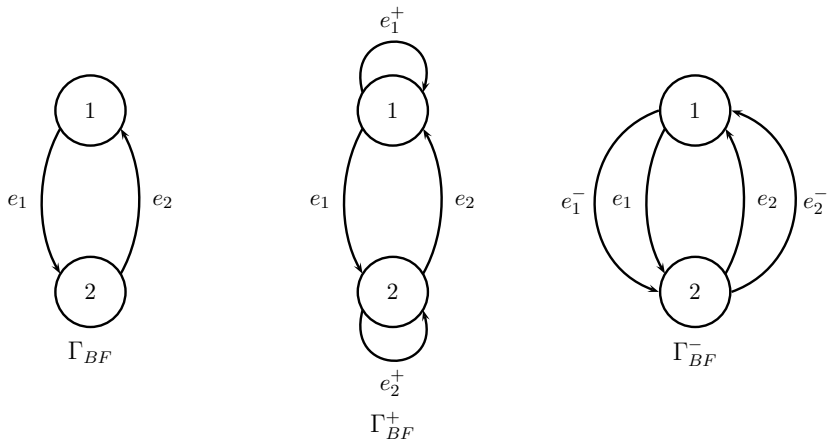


Figure 6.5 — Exemple de modèles modifiés

Définition 13. ***Produit cartésien de deux automates***
Le produit cartésien de deux automates $\Gamma_i =< S_i, S_{0i}, T_i, R_i >, i \in \{1,2\}$ *est l'automate* $\Gamma =< S, S_0, T, R >$ *défini par :*

- $S = S_1 \times S_2$;
- $S_0 = S_{01} \times S_{02}$;
- $T = T_1 \bigcup T_2 \bigcup T_1 \times T_2$;
- *R l'ensemble des arcs définis par* $((s_1, s_2), a, (s'_1, s'_2))$ *ssi :*
  - *soit* $(s_1, a, s'_1) \in R_1$ *alors* $s_2 = s'_2$;
  - *soit* $(s_2, a, s'_2) \in R_2$ *alors* $s_1 = s'_1$;
  - *soit* $a = (a_1, a_2)$ *et* $(s_1, a_1, s'_1) \in R_1$ *et* $(s_2, a_2, s'_2) \in R_2$;

Remarque : cela peut être noté également $a = (a, \epsilon)$ ou $a = (\epsilon, a)$, ce qui traduit la même chose que précédemment.

Définition 14. ***Produit synchronisé de deux automates***
Le produit synchronisé de deux automates Γ_1 *et* Γ_2*, notés* $\Gamma_1 \| \Gamma_2 \backslash Sync$ *est le produit cartésien restreint aux arcs labellisés dans* $Sync \subseteq T_1 \bigcup T_2 \bigcup (T_1 \times T_2)$.
Cet automate est noté $\Gamma =< S, S_0, Sync, R >$.

6.3.2 Les modèles synchronisés

Pour l'absence ou l'insertion d'événements, les produits synchronisés sont donnés par :

$$\Gamma^-_{Sync} = \Gamma^-_{BF} \| \Gamma^- \backslash Sync \text{ avec Sync} = \left\{ (e_i^-, e_i^-), e_i \right\}$$
$$\Gamma^+_{Sync} = \Gamma^+_{BF} \| \Gamma^+ \backslash Sync \text{ avec Sync} = \left\{ (e_i^+, e_i^+), e_i \right\}$$

Tous les paramètres définis par le produit synchronisé, notés $S^{(+/-)}_{sync}$, $S^{(+/-)}_{0sync}$, $T^{(+/-)}_{sync}$, $R^{(+/-)}_{sync}$ sont obtenus.

Le calcul du produit synchronisé entre les modèles $\Gamma^-_{BF} \| \Gamma^-$ sur les événements e_i^- fournit un modèle qui prend en compte l'ensemble des absences possibles dans le modèle de bon

comportement. Chaque état S_i du modèle normal peut donc être associé à une hypothèse de bon comportement (Ok) ou à une hypothèse d'absence ($F^-_{e_i}$). Ce produit synchronisé considère pour chaque état du modèle de bon comportement la possibilité que n'importe quel événement puisse être absent (représenté par un arc étiqueté par un événement non observable). Ainsi, lorsque le modèle se trouve dans un état (S_i,Ok), un arc r_i peut amener le marquage vers (S_{i+1},Ok) et un arc étiqueté par e_i^- vers un marquage (S_{i+1},$F^-_{e_i}$).
Nous considérons donc que tout événement associé aux arcs du modèle est susceptible d'être absent. Un modèle exhaustif de l'ensemble des possibilités d'absence dans le modèle est obtenu.

Le calcul du produit synchronisé entre les modèles $\Gamma^+_{BF}\|\Gamma^+$ sur les arcs étiquetés par e_i^+ fournit un modèle qui prend en compte l'ensemble des insertions possibles dans le modèle de bon comportement. Chaque état S_i du modèle normal peut donc être associé à une hypothèse de bon comportement (Ok) ou à une hypothèse d'insertion ($F^+_{e_i}$). Ce produit synchronisé considère pour chaque état du modèle de bon comportement la possibilité que n'importe quel événement attendu puisse s'insérer. Ainsi, lorsque le modèle se trouve dans un état (S_i,Ok), un arc r_i peut amener le marquage vers (S_{i+1},Ok) et un arc étiqueté par e_i^+ vers un marquage (S_i,$F^+_{e_i}$).
Nous considérons donc que tout événement associé aux arcs est susceptible de s'insérer à chaque état courant du modèle de bon comportement. Un modèle exhaustif de l'ensemble des possibilités d'insertion dans le modèle est obtenu.

Les deux modèles sont synchronisés sur les événements e_i qui ne se retrouvent que dans les modèles Γ^-_{BF} et Γ^+_{BF}. Les modèles Γ^-_{BF} et Γ^+_{BF} sont autorisés à évoluer dès la réception d'un événement alors que les modèles de classe de fautes restent dans leur état. Il y a également une synchronisation sur les événements e_i^- et e_i^+. Cela traduit le fait que lorsqu'un événement fautif se produit, alors les modèles Γ^-_{BF} et Γ^+_{BF} évoluent avec les modèles de classe de fautes Γ^- et Γ^+ qui passent d'un état Ok à un état $F^-_{t_i}$ ou $F^+_{t_i}$. Les modèles de classe de fautes ne peuvent pas évoluer sans que les modèles incluant des défaillances génériques évoluent également. Nous pourrions prendre en compte tous les événements, ce qui correspondrait à un produit cartésien, mais nous avons décidé de prendre en compte seulement les événements fautifs qui font évoluer le modèle de bon comportement. C'est pourquoi les couples d'événements $(\epsilon, e_i^{+/-})$ ne sont pas autorisés.

Le nombre des possibilités peut se traduire de la manière suivante :

soit n^s_{BF+} et n^t_{BF+} le nombre d'états et respectivement d'arcs du modèle Γ^+_{BF}. Le produit synchronisé $\Gamma^+_{BF}\|\Gamma^+ \setminus \left\{e_i, (e_i^+, e_i^+)\right\}$ comporte donc :

$$n^s_{SYNC+} = n^s_{BF+} \times n^s_{INS} = n^s_{BF} \times (n^t_{BF} + 1) \text{ états}$$

Il en est de même pour le produit synchronisé $\Gamma^-_{BF}\|\Gamma^- \setminus \left\{e_i, (e_i^-, e_i^-)\right\}$. La complexité ajoutée au modèle de bon comportement est donc une complexité polynomiale.

Le produit synchronisé est le point le plus délicat de cette approche puisqu'il génère un grand nombre d'états. Pour les modèles de classe de fautes, il y a $(n^t_{BF}+1)$ états et n^t_{BF} arcs. Les produits synchronisés génèrent quant à eux $n^s_{BF} \times (n^t_{BF}+1)$ états et au plus $(n^t_{BF} \times n^t_{BF})$ arcs. Cela peut entraîner pour des systèmes complexes une explosion d'états. Cependant,

puisque ces produits synchronisés sont exécutés hors-lignes, l'approche reste pertinente pour l'objectif de localisation.

6.3.3 Exemple

Reprenons l'exemple avec ses modèles de classes de fautes et ses modèles modifiés (figure 6.4 et figure 6.5). Les modèles synchronisés sont représentés par les automates de la figure 6.6.

L'automate synchronisé pour l'absence, noté $\Gamma^-_{Sync} = \Gamma^-_{BF} \| \Gamma^- \backslash Sync$ avec Sync $= \left\{(e_1^-, e_1^-), (e_1, \epsilon), (e_2^-, e_2^-), (e_2, \epsilon)\right\}$:

$$
\begin{array}{lcl}
\text{- } S^-_{sync} & = & S_1 \times S_2 = \{(1, Ok), (2, Ok), (1, F^-_{e_1}), (1, F^-_{e_2}), (2, F^-_{e_1}), (2, F^-_{e_2})\} \\
\text{- } S^-_{0sync} & = & S_{01} \times S_{02} = \{(1, Ok)\} \\
\text{- } T^-_{sync} & = & \left\{(e_1^-, e_1^-), (e_1, \epsilon), (e_2^-, e_2^-), (e_2, \epsilon)\right\} \\
\text{- } R^-_{sync} & = & \{((1, Ok), e_1, (2, Ok)), ((2, Ok), e_2, (1, Ok)), ((1, Ok), e_1^-, (2, F^-_{e_1})), \\
 & & ((2, Ok), e_2^-, (1, F^-_{e_2})), ((2, F^-_{e_1}), e_2, (1, F^-_{e_1})), ((1, F^-_{e_1}), e_1, (2, F^-_{e_1})), \\
 & & ((2, F^-_{e_2}), e_2, (1, F^-_{e_2})), ((1, F^-_{e_2}), e_1, (2, F^-_{e_2}))\}
\end{array}
$$

L'automate synchronisé pour l'insertion, noté $\Gamma^+_{Sync} = \Gamma^+_{BF} \| \Gamma^+ \backslash Sync$ avec Sync $= \left\{(e_1^+, e_1^+), (e_1, \epsilon), (e_2^+, e_2^+), (e_2, \epsilon)\right\}$:

$$
\begin{array}{lcl}
\text{- } S^+_{sync} & = & S_1 \times S_2 = \{(1, Ok), (2, Ok), (1, F^+_{e_1}), (1, F^+_{e_2}), (2, F^+_{e_1}), (2, F^+_{e_2})\} \\
\text{- } S^+_{0sync} & = & S_{01} \times S_{02} = \{(1, Ok)\} \\
\text{- } T^+_{sync} & = & \left\{(e_1^+, e_1^+), (e_1, \epsilon), (e_2^+, e_2^+), (e_2, \epsilon)\right\} \\
\text{- } R^+_{sync} & = & \{((1, Ok), e_1, (2, Ok)), ((2, Ok), e_2, (1, Ok)), ((1, Ok), e_1^+, (1, F^+_{e_1})), \\
 & & ((2, Ok), e_2^+, (2, F^+_{e_2})), ((2, F^+_{e_1}), e_2, (1, F^+_{e_1})), ((1, F^+_{e_1}), e_1, (2, F^+_{e_1})), \\
 & & ((2, F^+_{e_2}), e_2, (1, F^+_{e_2})), ((1, F^+_{e_2}), e_1, (2, F^+_{e_2}))\}
\end{array}
$$

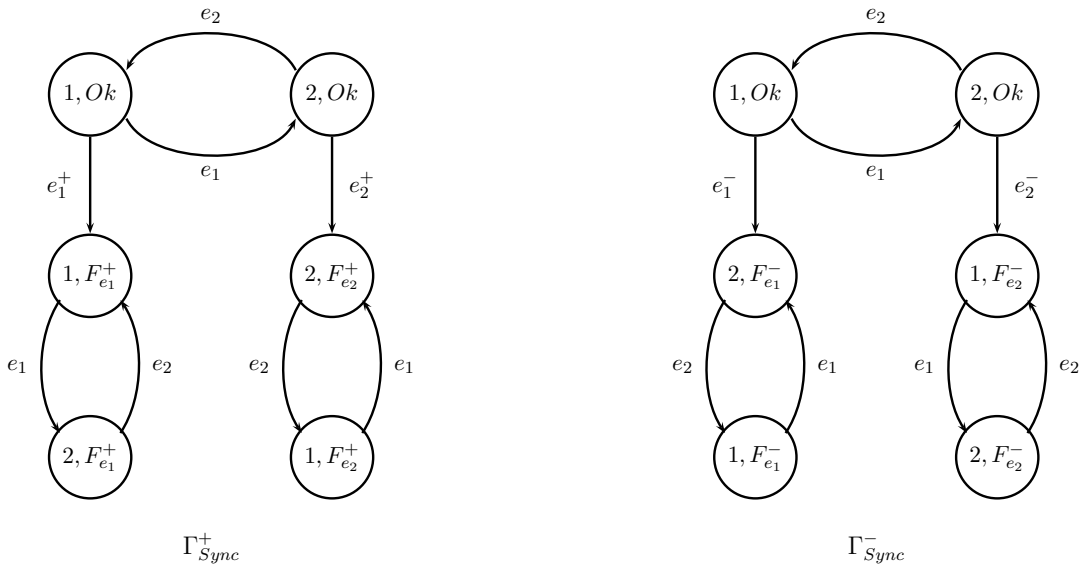


Figure 6.6 — Exemple : modèles synchronisés

6.4 Les Diagnostiqueurs

À partir de ces modèles Γ^{+}_{Sync} et Γ^{+}_{Sync} où sont représentés toutes les possibilités, il faut prendre en considération les significations des arcs étiquetés par e_i^+ et e_i^-. De plus, sur ces modèles sont représentés des états qui ne seront jamais atteints parce qu'ils font parti d'une trajectoire issue d'une défaillance que nous ne voulons pas suivre. La détection est effectuée plus en amont et par conséquent le système n'atteindra jamais ces états puisqu'ils ne correspondent pas au comportement normal. Deux automates appelés diagnostiqueurs sont construits (l'un pour l'insertion et l'autre pour l'absence) dans lesquels sont compilées toutes les informations de diagnostic. Ces deux diagnostiqueurs sont notés $\Gamma^{-}_{Diag} =< S^{-}_{Diag}, S^{-}_{0_{Diag}}, T^{-}_{Diag}, R^{-}_{Diag} >$ et $\Gamma^{+}_{Diag} =< S^{+}_{Diag}, S^{+}_{0_{Diag}}, T^{+}_{Diag}, R^{+}_{Diag} >$.

6.4.1 Absence d'un événement : construction de Γ^{-}_{Diag}

Un arc étiqueté par e_i^- correspond à l'absence d'un événement et est donc associé à un événement non observable (e_i^-). Aussi, d'un point de vue observation, s'il existe (s_j, e_i^-, s'_j), les états s_j et s'_j ne peuvent pas être discriminés et sont donc regroupés ensemble dans Γ^{-}_{Diag}. Il existe une ambiguïté entre les deux états.

Pour prendre en compte cette caractéristique, le mécanisme de fusion d'états est à nouveau utilisé (cf chapitre 4.1.2.1). Il faut donc caractériser la partition π (cf définition 6). Dans une première étape, les états qui ne sont reliés par aucun événement observable sont rassemblés. Une nouvelle partition de l'ensemble d'états est obtenue qui permet de construire un automate noté Γ^{-}_{Diag0} à l'aide de la définition 7. Cependant, avec la fusion d'états définie précédemment, le modèle obtenu conserve encore les arcs dont les éléments sont non observables. Comme nous ne voulons conserver sur le modèle que les arcs associés à des éléments observables, ces arcs sont supprimés. Cela n'a aucune incidence sur la structure de l'automate dans la mesure où par construction, pour tout arc (s_j, e_i^-, s'_j), $s_j = s'_j$ et donc seul un arc qui a pour état de sortie son état d'entrée est supprimé. Au final, le résultat constitue encore un automate . Les étiquettes e^- sont supprimées de l'ensemble T^{-}_{Diag0}.

Après avoir supprimé du modèle les événements non observables, il faut, dans une seconde étape, vérifier s'il existe deux arcs provenant du même état ayant même étiquette (i.e. même événement). Si c'est le cas alors il faut fusionner ces états. Par conséquent, une partition π est à nouveau construite. Une nouvelle partition de l'ensemble d'états S^{-}_{Diag0} est obtenu. Cette partition permet d'avoir un automate par fusion d'états noté Γ^{-}_{Diag}. Il faut également supprimer tous les arcs redondants dans R^{-}_{Diag}.

Tant qu'il existe encore des arcs ayant même état d'entrée et de sortie alors les deux étapes ci-dessus sont réappliquées.

Pour finir, dans une troisième et dernière étape, puisqu'une détection a lieu dès la sortie du bon comportement, la construction du diagnostiqueur est arrêtée dès qu'il n'existe plus que des hypothèses de fautes dans les états. Pour cela, les arcs qui ont pour état d'entrée seulement des hypothèses de fautes sont supprimés. Rappelons que S^{-}_{diag} est finalement une partition de S^{-}_{sync} et donc que chaque élément de S^{-}_{diag} est constitué d'éléments de S^{-}_{sync}, i.e. de couples de $S^{-}_{BF} \times S^{-}$. Puisque des arcs sont supprimés, il faut supprimer tous les états

qui ne sont plus reliés par des arcs.

Cette procédure est décrite formellement ci-dessous.

Étape 1 :

1. $\pi = \{\{S_i\}, \forall S_i \in S^-_{sync}\}$,
2. $\forall r = (q, e, q') \in R^-_{sync}$ *si* $e \notin E_{obs}$ alors
   $\pi = \pi \cup \{\{B(q,\pi) \cup B(q',\pi)\}\} \setminus \{\{B(q,\pi)\}, \{B(q',\pi)\}\}$.
3. $\Gamma^-_{Diag0} = \Gamma^-_{Sync}/\pi =< S^-_{Diag0}, B(S^-_{0_{Sync}}, \pi), T^-_{Diag0}, R^-_{Diag0} >$, avec :
   - $S^-_{Diag0} = \{B(s,\pi) | s \in S^-_{Sync}\}$,
   - $T^-_{Diag0} = T^-_{Sync}$,
   - $R^-_{Diag0} : S^-_{Diag0} \times T^-_{Diag0} \times S^-_{Diag0} : \forall B, B' \in S^-_{Diag0}, \forall t \in T^-_{Diag0}, (B, t, B') \in R^-_{Diag0}$ si et seulement si $\exists q, q' \in S^-_{Sync}, q \in B, q' \in B'$ *et* $(q, t, q') \in R^-_{Sync}$.
4. $\forall r' = (q, e, q') \in R^-_{Diag0}$ *si* $e \notin E_{obs}$ alors
   $R^-_{Diag0} = R^-_{Diag0} \setminus \{r'\}$,
5. $\forall e^- \in T^-_{Diag0}, T^-_{Diag0} = T^-_{Diag0} \setminus \{e^-\}$.

Étape 2 :

1. $\pi = \{\{S_i\}, \forall S_i \in S^-_{Diag0}\}$,
2. *Si* $\exists r = (q, e, q')$ et $r' = (q, e, q'') \in R^-_{Diag0}$ alors
   $\pi = \pi \cup \{\{B(q',\pi) \cup B(q'',\pi)\}\} \setminus \{\{B(q',\pi)\}, \{B(q'',\pi)\}\}$,
3. $\Gamma^-_{Diag} = \Gamma^-_{Diag0}/\pi =< S^-_{Diag}, B(S^-_{0_{Diag0}}, \pi), T^-_{Diag}, R^-_{Diag} >$, avec :
   - $S^-_{Diag} = \{B(s,\pi) | s \in S^-_{Diag0}\}$,
   - $T^-_{Diag} = T^-_{Diag0}$,
   - $R^-_{Diag} : S^-_{Diag} \times T^-_{Diag} \times S^-_{Diag} : \forall B, B' \in S^-_{Diag}, \forall t \in T^-_{Diag}, (B, t, B') \in R^-_{Diag}$ si et seulement si $\exists q, q' \in S^-_{Diag0}, q \in B, q' \in B'$ *et* $(q, t, q') \in R^-_{Diag0}$.
4. *Si* $\exists r = (q, e, q')$ *et* $r' = (q, e, q') \in R^-_{diag}$ alors
   $R^-_{diag} = R^-_{diag} \setminus \{r'\}$.

À partir de ce nouvel automate, la procédure précédente est réappliquée tant que l'automate comporte encore des arcs ayant même état d'entrée et même étiquette.

Étape 3 :

1. *Si* $\exists r = (q, e, q') \in R^-_{diag} | \forall a \in q, Ok \notin a$, alors
   $R^-_{diag} = R^-_{diag} \setminus \{r\}$,
2. $\forall q \in S^-_{diag}$, *si* $\nexists r \in R^-_{diag} | r = (q, e, q')$ ou $r = (q', e, q)$ alors
   $S^-_{diag} = S^-_{diag} \setminus \{q\}$.

Ceci termine la construction du diagnostiqueur. Notons que cette dernière règle peut être modifiée si nous souhaitons conserver plus d'information. Par exemple, s'il existe plusieurs hypothèses de fautes dans un même état, alors la construction du diagnostiqueur pourrait être poursuivie pour n'avoir plus qu'une seule hypothèse de faute. Cet aspect de la construction du diagnostiqueur rejoint la problématique de la diagnosticabilité du système.

6.4.2 Insertion d'un événement : construction de Γ^+_{Diag}

Dans le modèle Γ^+_{Sync}, une étiquette e^+ correspond en terme d'observation à l'étiquette e. Ce sont les mêmes événements. Si les arcs ont pour étiquettes e^+ et e ainsi que les mêmes états d'entrée, alors lors de l'occurrence d'un événement, il n'est pas possible de savoir quel état est atteint. Il y a une ambiguïté. Il faut donc fusionner les états de sortie des arcs étiqueté par e et e^+ puisqu'ils sont associés au même événement.

Dans une première étape, il faut vérifier s'il existe deux arcs provenant du même état ayant comme étiquette e et e^+. Si c'est le cas alors il faut fusionner les états de sorties de ces arcs. Par conséquent, une partition π est construite qui permet de construire l'automate par fusion d'états à l'aide de la définition 7 et qui est noté Γ^+_{Diag0}. Puis il faut supprimer toutes les arcs redondants dans R^+_{Diag0} et les étiquettes e^+ de l'ensemble T^+_{Diag0}.

Dans une seconde étape, après avoir fusionné des états du modèle, il faut vérifier s'il existe deux arcs provenant du même état et ayant même étiquette. Si c'est le cas alors il faut fusionner ces états. Par conséquent, une partition π est définie permettant d'avoir à nouveau un automate par fusion d'états, noté Γ^+_{Diag}.
Il faut supprimer tous les arcs redondants dans R^+_{diag}.

Comme pour l'absence, dans une troisième étape, la construction du diagnostiqueur est stoppée dès qu'il n'y a plus que des hypothèses de fautes dans les états. Pour cela, les arcs qui ont pour état d'entrée seulement des hypothèses de fautes sont supprimés. Puisque des arcs sont supprimés, il faut maintenant supprimer tous les états qui ne sont plus reliés par des arcs.

Étape 1 :

1. $\pi = \{\{S_i\}, \forall S_i \in S^+_{sync}\}$,
2. $Si\ \exists r = (q, e, q')$ et $r' = (q, e^+, q'') \in R^+_{sync}$ alors
   $\pi = \pi \cup \{\{B(q', \pi) \cup B(q'', \pi)\}\} \setminus \{\{B(q', \pi)\}, \{B(q'', \pi)\}\}$,
3. $\Gamma^+_{Diag0} = \Gamma^+_{Sync}/\pi =< S^+_{Diag0}, B(S^+_{0_{Sync}}, \pi), T^+_{Diag0}, R^+_{Diag0} >$, avec :
   - $S^+_{Diag0} = \{B(s, \pi) | s \in S^+_{Sync}\}$,
   - $T^+_{Diag0} = T^+_{Sync}$,
   - $R^+_{Diag0} : S^+_{Diag0} \times T^+_{Diag0} \times S^+_{Diag0} : \forall B, B' \in S^+_{Diag0}, \forall t \in T^+_{Diag0}, (B, t, B') \in R^+_{Diag0}$ si et seulement si $\exists q, q' \in S^+_{Sync}, q \in B, q' \in B'\ et\ (q, t, q') \in R^+_{Sync}$.
4. $Si\ \exists r = (q, e, q')\ et\ r' = (q, e^+, q') \in R^+_{Diag0}$ alors
   $R^+_{Diag0} = R^+_{Diag0} \setminus \{r'\}$,
5. $\forall e^+ \in T^+_{Diag0}, T^+_{Diag0} = T^+_{Diag0} \setminus \{e^+\}$.

Étape 2 :

1. $\pi = \{\{S_i\}, \forall S_i \in S^+_{diag}\}$,
2. $Si\ \exists r = (q, e, q')\ et\ r' = (q, e, q'') \in R^+_{Diag0}$ alors
   $\pi = \pi \cup \{\{B(q', \pi) \cup B(q'', \pi)\}\} \setminus \{\{B(q', \pi)\}, \{B(q'', \pi)\}\}$,
3. $\Gamma^+_{Diag} = \Gamma^+_{Diag0}/\pi =< S^+_{Diag}, B(S^+_{0_{Diag0}}, \pi), T^+_{Diag}, R^+_{Diag} >$, avec :
   - $S^+_{Diag} = \{B(s, \pi) | s \in S^+_{Diag0}\}$,

- $T^+_{Diag} = T^+_{Diag0}$,
- $R^+_{Diag} : S^+_{Diag} \times T^+_{Diag} \times S^+_{Diag} : \forall B, B' \in S^+_{Diag}, \forall t \in T^+_{Diag}, (B, t, B') \in R^+_{Diag}$ si et seulement si $\exists q, q' \in S^+_{Diag0}, q \in B, q' \in B'$ *et* $(q, t, q') \in R^+_{Diag0}$.

4. *Si* $\exists r = (q, e, q')$ *et* $r' = (q, e, q') \in R^+_{diag}$ alors
   $R^+_{diag} = R^+_{diag} \backslash \{r'\}$.

À partir de ce nouvel automate, la procédure précédente est poursuivie tant que l'automate comporte encore des arcs ayant même état d'entrée et même étiquette.

Étape 3 :

1. *Si* $\exists r = (q, e, q') \in R^+_{diag} | \forall a \in q, Ok \notin a$, alors
   $R^+_{diag} = R^+_{diag} \backslash \{r\}$,
2. $\forall q \in S^+_{diag}, Si \quad \nexists r \in R^+_{diag} | r = (q, e, q')$ *ou* $r = (q', e, q)$ alors
   $S^+_{diag} = S^+_{diag} \backslash \{q\}$.

Ceci termine la construction du diagnostiqueur. De la même façon, cette dernière règle peut être modifiée si nous souhaitons ajouter plus de précision.

6.4.3 Exemple

Les diagnostiqueurs sont construits à partir des différents automates précédemment, en suivant les étapes définies ci-dessus.

6.4.3.1 Diagnostiqueur absence :

Soit l'automate Γ^-_{Sync} :

- $S^-_{sync} = S_1 \times S_2 = \{(1, Ok), (2, Ok), (1, F^-_{e_1}), (1, F^-_{e_2}), (2, F^-_{e_1}), (2, F^-_{e_2})\}$
- $S^-_{0sync} = S_{01} \times S_{02} = \{(1, Ok)\}$
- $T^-_{sync} = \left\{(e^-_1, e^-_1), (e_1, \epsilon), (e^-_2, e^-_2), (e_2, \epsilon)\right\}$
- $R^-_{sync} = \{((1, Ok), e_1, (2, Ok)), ((2, Ok), e_2, (1, Ok)), ((1, Ok), e^-_1, (2, F^-_{e_1})),$
  $((2, Ok), e_1 2^-, (1, F^-_{e_2})), ((2, F^-_{e_1}), e_2, (1, F^-_{e_1})), ((1, F^-_{e_1}), e_1, (2, F^-_{e_1})),$
  $((2, F^-_{e_2}), e_2, (1, F^-_{e_2})), ((1, F^-_{e_2}), e_1, (2, F^-_{e_2}))\}$

Étape 1 : Définissons la partition π par :

$$\pi = \{\{(1, Ok)\}, \{(2, Ok)\}, \{(1, F^-_{e_1})\}, \{(1, F^-_{e_2})\}, \{(2, F^-_{e_1})\}, \{(2, F^-_{e_2})\}\}$$

Il existe deux arcs dont l'étiquette ne fait pas partie des événements observables :

$$\{((1, Ok), e^-_1, (2, F^-_{e_1})), ((2, Ok), e^-_2, (1, F^-_{e_2}))\}$$

La nouvelle partition devient :

$$\begin{aligned}\pi = \quad & \pi \cup \{\{(1,Ok),(2,F^-_{e_1})\},\{(2,Ok),(1,F^-_{e_2})\}\} \\ & \backslash\{\{(1,Ok)\},\{(2,F^-_{e_1})\},\{(2,Ok)\},\{(1,F^-_{e_2})\}\} \\ \pi = \quad & \{\{(1,F^-_{e_1})\},\{(2,F^-_{e_2})\},\{(1,Ok),(2,F^-_{e_1})\},\{(2,Ok),(1,F^-_{e_2})\}\}\end{aligned}$$

L'automate correspondant est ensuite construit :

$$\begin{aligned}S^-_{diag} = \quad & \{\{(1,F^-_{e_1})\},\{(2,F^-_{e_2})\},\{(1,Ok),(2,F^-_{e_1})\},\{(2,Ok),(1,F^-_{e_2})\}\} \\ S^-_{0diag} = \quad & \{\{(1,Ok),(2,F^-_{e_1})\}\} \\ T^-_{diag} = \quad & \left\{(e_1^-,e_1^-),(e_1,\epsilon),(e_2^-,e_2^-),(e_2,\epsilon)\right\} \\ R^-_{diag} = \quad & (\{(1,Ok),(2,F^-_{e_1})\},e_1,\{(2,Ok),(1,F^-_{e_2})\}), \\ & (\{(2,Ok),(1,F^-_{e_2})\},e_2,\{(1,Ok),(2,F^-_{e_1})\}), \\ & (\{(1,Ok),(2,F^-_{e_1})\},e_1^-,\{(1,Ok),(2,F^-_{e_1})\}, \\ & (\{(2,Ok),(1,F^-_{e_2})\},e_2^-,\{(2,Ok),(1,F^-_{e_2})\}), \\ & (\{(1,Ok),(2,F^-_{e_1})\},e_2,\{(1,F^-_{e_1})\}), \\ & (\{(1,F^-_{e_1})\},e_1,\{(1,Ok),(2,F^-_{e_1})\}),, \\ & (\{(2,F^-_{e_2})\},e_2,\{(2,Ok),(1,F^-_{e_2})\}), \\ & (\{(2,Ok),(1,F^-_{e_2})\},e_1,\{(2,F^-_{e_2})\})\}\end{aligned}$$

À présent, supprimons les arcs qui ont des étiquettes appartenant à l'ensemble des événements non observables. L'ensemble $R^-_{diag} =$ devient :

$$\begin{aligned}R^-_{diag} = \quad & (\{(1,Ok),(2,F^-_{e_1})\},e_1,\{(2,Ok),(1,F^-_{e_2})\}), \\ & (\{(2,Ok),(1,F^-_{e_2})\},e_2,\{(1,Ok),(2,F^-_{e_1})\}), \\ & (\{(1,Ok),(2,F^-_{e_1})\},e_2,\{(1,F^-_{e_1})\}), \\ & (\{(1,F^-_{e_1})\},e_1,\{(1,Ok),(2,F^-_{e_1})\}), \\ & (\{(2,F^-_{e_2})\},e_2,\{(2,Ok),(1,F^-_{e_2})\}), \\ & (\{(2,Ok),(1,F^-_{e_2})\},e_1,\{(2,F^-_{e_2})\})\}\end{aligned}$$

Etape 2 : il n'existe pas dans l'ensemble R^-_{diag} d'arc r et r' tels que $r = (q,e,q')$ et $r' = (q,e,q'')$.

Etape 3 : dans cette étape, les arcs qui proviennent uniquement d'hypothèses de fautes sont supprimés. Il y a deux arcs correspondant à cette définition :

$$(\{(1,F^-_{e_1})\},e_1,\{(1,Ok),(2,F^-_{e_1})\}), \text{ et } (\{(2,F^-_{e_2})\},e_2,\{(2,Ok),(1,F^-_{e_2})\})$$

L'ensemble R^-_{diag} devient :

$$\begin{aligned}R^-_{diag} = \quad & (\{(1,Ok),(2,F^-_{e_1})\},e_1,\{(2,Ok),(1,F^-_{e_2})\}), \\ & (\{(2,Ok),(1,F^-_{e_2})\},e_2,\{(1,Ok),(2,F^-_{e_1})\}), \\ & (\{(1,Ok),(2,F^-_{e_1})\},e_2,\{(1,F^-_{e_1})\}), \\ & (\{(2,Ok),(1,F^-_{e_2})\},e_1,\{(2,F^-_{e_2})\})\}\end{aligned}$$

Il n'existe pas d'état seul sans arc en entrée ou en sortie.

À partir de cet ensemble d'arcs R^-_{diag} et des ensembles T^-_{diag}, S^-_{0diag}, et S^-_{diag} définis précédemment, le diagnostiqueur obtenu est représenté sur la figure 6.7.

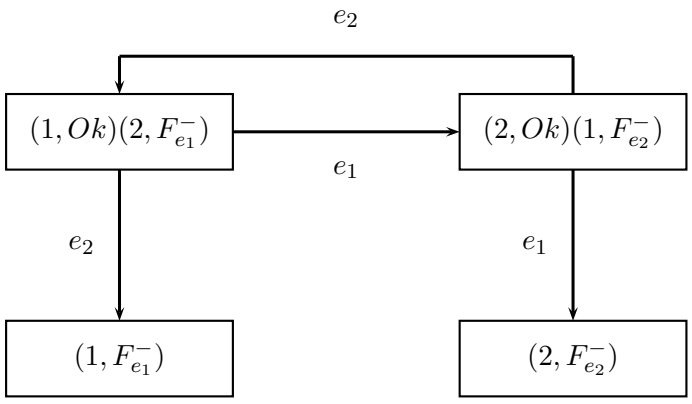


Figure 6.7 — Exemple : Diagnostiqueur absence

6.4.3.2 Diagnostiqueur insertion :

Soit l'automate Γ^+_{Sync} :

- $S^+_{sync} = S_1 \times S_2 = \{(1, Ok), (2, Ok), (1, F^+_{e_1}), (1, F^+_{e_2}), (2, F^+_{e_1}), (2, F^+_{e_2})\}$
- $S^+_{0sync} = S_{01} \times S_{02} = \{(1, Ok)\}$
- $T^+_{sync} = \left\{(e^+_1, e^+_1), (e_1, \epsilon), (e^+_2, e^+_2), (e_2, \epsilon)\right\}$
- $R^+_{sync} = \{((1, Ok), e_1, (2, Ok)), ((2, Ok), e_2, (1, Ok)), ((1, Ok), e^+_1, (1, F^+_{e_1})),$
  $((2, Ok), e^+_2, (2, F^+_{e_2})), ((2, F^+_{e_1}), e_2, (1, F^+_{e_1})), ((1, F^+_{e_1}), e_1, (2, F^+_{e_1})),$
  $((2, F^+_{e_2}), e_2, (1, F^+_{e_2})), ((1, F^+_{e_2}), e_1, (2, F^+_{e_2}))\}$

Étape 1 : comme ci-dessus, définissons la partition π par :

$$\pi = \{\{(1, Ok)\}, \{(2, Ok)\}, \{(1, F^+_{e_1})\}, \{(1, F^+_{e_2})\}, \{(2, F^+_{e_1})\}, \{(2, F^+_{e_2})\}\}$$

Il y a ici deux arcs r et r' de R^+_{sync} tels que $r = (q, e, q')$ et $r' = (q, e^+, q'')$. Il s'agit de :

$$\left\{((1, Ok), e_1, (2, Ok)), ((2, Ok), e_2, (1, Ok)), ((1, Ok), e^+_1, (1, F^+_{e_1})), ((2, Ok), e^+_2, (2, F^+_{e_2}))\right\}$$

La nouvelle partition obtenue est :

$$\pi = \{\{(1, Ok)(2, F^+_{e_2})\}, \{(2, Ok)(1, F^+_{e_1})\}, \{(1, F^+_{e_2})\}, \{(2, F^+_{e_1})\}\}$$

L'automate correspondant est ensuite construit :

$$
\begin{aligned}
S^+_{diag} = \quad & \{\{(1,Ok)(2,F^+_{e_2})\},\{(2,Ok)(1,F^+_{e_1})\},\{(1,F^+_{e_2})\},\{(2,F^+_{e_1})\}\} \\
S^+_{0diag} = \quad & \{\{(1,Ok),(2,F^+_{e_2})\}\} \\
T^+_{diag} = \quad & \left\{(e^+_1,e^+_1),(e_1,\epsilon),(e^+_2,e^+_2),(e_2,\epsilon)\right\} \\
R^+_{diag} = \quad & \{(\{(1,Ok),(2,F^+_{e_2})\},e_1,\{(2,Ok),(1,F^+_{e_1})\}), \\
& (\{(2,Ok),(1,F^+_{e_1})\},e_2,\{(1,Ok),(2,F^+_{e_2})\}), \\
& (\{(1,Ok),(2,F^+_{e_2})\},e^+_1,\{(2,Ok),(1,F^+_{e_1})\}, \\
& (\{(2,Ok),(1,F^+_{e_1})\},e^+_2,\{(1,Ok),(2,F^+_{e_2})\}), \\
& (\{(1,Ok),(2,F^+_{e_2})\},e_2,\{(1,F^+_{e_2})\}), \\
& (\{(1,F^+_{e_2})\},e_1,\{(1,Ok),(2,F^+_{e_2})\}), \\
& (\{(2,F^+_{e_1})\},e_2,\{(2,Ok),(1,F^+_{e_1})\}), \\
& (\{(2,Ok),(1,F^+_{e_1})\},e_1,\{(2,F^+_{e_1})\})\}
\end{aligned}
$$

Il existe dans R^+_{diag} des arcs redondants :
$(\{(1,Ok),(2,F^+_{e_2})\},e_1,\{(2,Ok),(1,F^+_{e_1})\})$ et $(\{(1,Ok),(2,F^+_{e_2})\},e^+_1,\{(2,Ok),(1,F^+_{e_1})\})$
$(\{(2,Ok),(1,F^+_{e_1})\},e_2,\{(1,Ok),(2,F^+_{e_2})\})$ et $(\{(2,Ok),(1,F^+_{e_1})\},e^+_2,\{(1,Ok),(2,F^+_{e_2})\})$

Les arcs redondants sont supprimés. R^+_{diag} devient :

$$
\begin{aligned}
R^+_{diag} = \quad & \{(\{(1,Ok),(2,F^+_{e_2})\},e_1,\{(2,Ok),(1,F^+_{e_1})\}), \\
& (\{(2,Ok),(1,F^+_{e_1})\},e_2,\{(1,Ok),(2,F^+_{e_2})\}), \\
& (\{(1,Ok),(2,F^+_{e_2})\},e_2,\{(1,F^+_{e_2})\}), \\
& (\{(1,F^+_{e_2})\},e_1,\{(1,Ok),(2,F^+_{e_2})\}), \\
& (\{(2,F^+_{e_1})\},e_2,\{(2,Ok),(1,F^+_{e_1})\}), \\
& (\{(2,Ok),(1,F^+_{e_1})\},e_1,\{(2,F^+_{e_1})\})\}
\end{aligned}
$$

Etape 2 : il n'y a plus d'arcs dans l'automate ayant même état d'entrée et même étiquette. Par conséquent, nous pouvons passer à l'étape 3.

Etape 3 : dans cette étape, les arcs qui proviennent uniquement d'hypothèses de fautes sont supprimés. Il y a deux arcs correspondant à cette définition :

$$(\{(1,F^+_{e_2})\},e_1,\{(1,Ok),(2,F^+_{e_2})\}) \text{ et } (\{(2,F^+_{e_1})\},e_2,\{(2,Ok),(1,F^+_{e_1})\})$$

L'ensemble R^+_{diag} devient :

$$
\begin{aligned}
R^+_{diag} = \quad & \{(\{(1,Ok),(2,F^+_{e_2})\},e_1,\{(2,Ok),(1,F^+_{e_1})\}), \\
& (\{(2,Ok),(1,F^+_{e_1})\},e_2,\{(1,Ok),(2,F^+_{e_2})\}), \\
& (\{(1,Ok),(2,F^+_{e_2})\},e_2,\{(1,F^+_{e_2})\}), \\
& (\{(2,Ok),(1,F^+_{e_1})\},e_1,\{(2,F^+_{e_1})\})\}
\end{aligned}
$$

Il n'existe pas d'état seul sans arc en entrée ou en sortie.

À partir de cet ensemble d'arcs R^+_{diag} et des ensembles T^+_{diag}, S^+_{0diag}, et S^+_{diag} définis précédemment, le diagnostiqueur obtenu est représenté sur la figure 6.8.

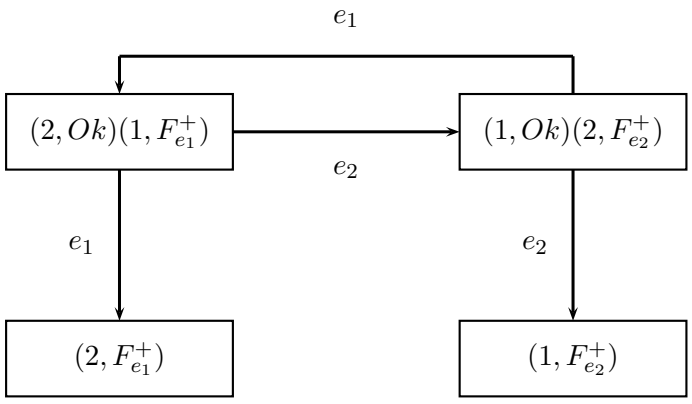


Figure 6.8 — Exemple : Diagnostiqueur insertion

6.5 Conclusion

Dans ce chapitre, nous avons établi une approche qui permet de localiser un événement qui s'est inséré ou qui est manquant. Cette approche utilise seulement les connaissances du système à savoir son comportement normal.

L'approche présentée résout le problème posé dans le chapitre précédent et qui concernait le nombre des calculs en ligne. Ici, la surveillance et la localisation se fait par l'intermédiaire d'un automate dans lequel l'information de diagnostic est compilée. Il suffit donc de suivre en ligne le diagnostiqueur et une fois qu'un événement incohérent est détecté, l'état courant de l'automate permet d'émettre des hypothèses d'insertion ou d'absence d'un événement. Ceci permet de faire un minimum de calculs en ligne (juste un suivi en ligne).

La construction du diagnostiqueur se fait à partir du modèle à événements discrets représentant le bon comportement du système. Divers modèles sont construits de façon automatique qui sont ensuite utilisés pour obtenir au final le diagnostiqueur.

En outre, dans le chapitre précédent, nous avions exposé un problème de détectabilité qui ne permettait pas à l'algorithme de converger. En utilisant l'approche diagnostiqueur, le problème de convergence est résolu. La détectabilité d'une défaillance lorsque le système est dans un état particulier peut être étudiée directement à partir du diagnostiqueur. Si, par exemple, dans un même état, deux couples sont associés comme par exemple (1,Ok) et (1,$F^{+/-}_{e_i}$), i.e. un même état (ici l'état 1) associé à une hypothèse de bon comportement et à une hypothèse de faute, alors il n'est pas possible de détecter l'insertion ou l'absence de l'événement e_i car il suit le comportement normal du système.

Le problème qui se pose ici se situe au niveau de la construction du diagnostiqueur. En effet, le produit synchronisé effectué entre les modèles de classe de fautes et le modèle de bon comportement modifié peut amener à une explosion d'états si la taille du modèle est grande. Nous avons donc décidé de mettre en place une nouvelle approche, toujours basée sur le principe du diagnostiqueur, mais qui utilise le formalisme des réseaux de Petri. Le chapitre

suivant est consacré à la présentation de cette nouvelle approche.

7 Approche diagnostiqueur : les réseaux de Petri

Introduction

Dans le chapitre précédent (chapitre 6), nous avons vu comment construire un modèle prenant en compte l'ensemble des insertions et des absences d'un événement. Ce modèle est construit à partir du modèle de bon comportement du système. La technique de détection et de localisation en ligne consiste alors à suivre l'évolution des modèles construits, ce qui demande peu de calculs. Néanmoins, la construction des modèles utilise un produit synchronisé, ce qui pour des systèmes complexes pose le problème de l'explosion combinatoire du nombre des états.

Nous avons donc voulu dans ce chapitre, conserver l'avantage de l'approche diagnostiqueur tout en essayant de résoudre le problème du produit synchronisé. Pour cela, une méthode a été développée qui utilise le formalisme des réseaux de Petri. En effet, il est possible en s'appuyant sur la puissance de représentation graphique des réseaux de Petri de synchroniser les modèles, sans pour autant réaliser le produit synchronisé.

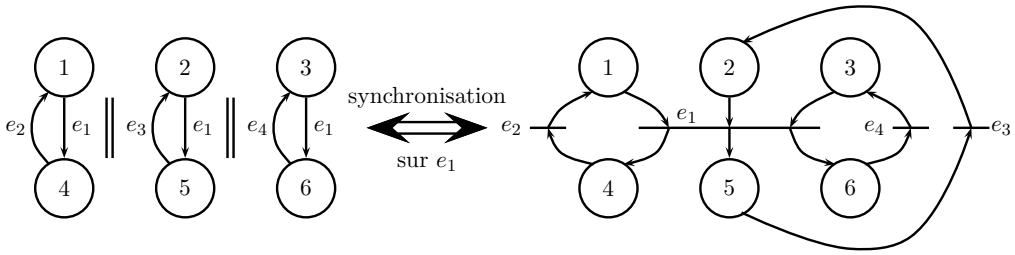


Figure 7.1 — Principe de synchronisation

Par exemple, sur la figure 7.1, si nous synchronisons les trois modèles sur l'événement e_1, le produit synchronisé des automates donnera onze places et dix-sept arcs. La synchronisation des modèles sur le réseaux de Petri ne donne que six places. Nous avons seulement une somme des places du réseau de Petri contrairement au produit synchronisé. Le produit synchronisé des automates se retrouvent lorsque le graphe d'accessibilité du réseau de Petri est établi. L'avantage est qu'il n'est pas nécessaire de construire le produit synchronisé pour obtenir un diagnostiqueur (comme pour les automates). La construction du graphe d'accessibilité est

modifiée, ce qui évite la construction de ce produit synchronisé.

Le principe de construction du diagnostiqueur en réseau de Petri (figure 7.2) est similaire à celui à base d'automate. En effet, à partir d'un réseau de Petri de bon comportement noté $R_{BF} =< P_{BF}, T_{BF}, I_{BF}, O_{BF}, M_{0-BF} >$, nous allons construire les modèles de classes de fautes et les modèles de bon comportement incluant les fautes génériques. À la place des modèles des produits synchronisés sont construits des modèles représentant la synchronisation. Enfin, les modèles des diagnostiqueurs sont obtenus à partir des graphes d'accessibilité des réseaux de Petri (Soldani *et al.*, 2007*b*).

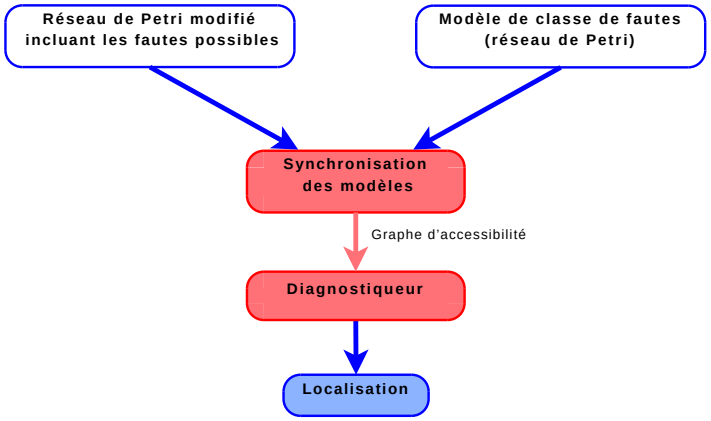


Figure 7.2 — Principe du diagnostic à base de modèles à l'aide du diagnostiqueur

7.1 Modèles de Classe de Fautes

7.1.1 Le modèle

De la même manière que pour les automates, des modèles de classes de fautes sont construits et sont prsésentés sous le formalisme des réseaux de Petri sur la figure 7.3.

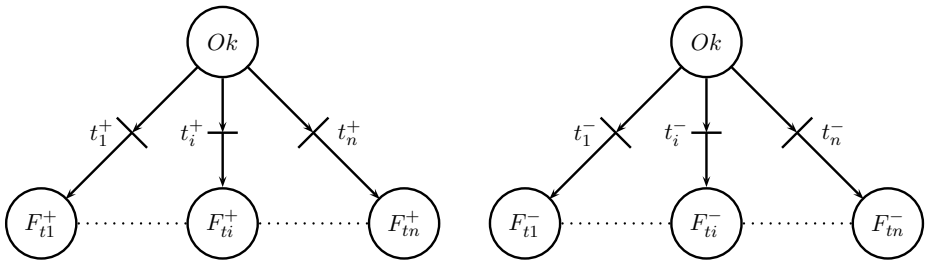


Figure 7.3 — Modèles de classe de fautes (insertion et absence respectivement)

Le modèle représente le passage d'un mode de bon comportement (OK) vers un mode d'hypothèse de "faute" $F_{t_i}^+$ ou $F_{t_i}^-$. Ces hypothèses représentent respectivement l'insertion ou à l'absence de l'événement associé à la transition t_i. Considérer la transition et non l'événement apporte une information supplémentaire sur le moment où la défaillance a eu lieu. En effet,

un événement peut être associé à différentes transitions. Si cette transition est connue, alors l'état dans lequel le modèle se trouvait avant de recevoir cet événement fautif est connu. Cela apporte un complément d'information en plus de la connaissance de l'événement fautif.

Ces modèles correspondent exactement à ceux présentés dans le chapitre précédent mais basé réseaux de Petri. Ces modèles sont notés R^+ et R^-. Le modèle de bon comportement est noté R_{BF}. Les labels associés aux marquages sont $\left\{Ok, F^+_{t_i}, i \in [1, \ldots, n^t_{BF}]\right\}$.

Le réseau de Petri $R^+ =< P^+, T^+, I^+, O^+, M^+_0 >$ est défini par :
- $P^+ = \left\{p^+_i, i \in [1, \ldots, n^t_{BF} + 1]\right\}$;
- M^+_0 est défini par $M^+_0(p^+_1) = 1$ et $M^+_0(p^+_i) = 0, i \in [2, n^t_{BF} + 1]$ tel que $l(M^+_0) = Ok$;
- $T^+ = \left\{t^+_i, i \in [1, \ldots, n^t_{BF}] | E_t(t^+_i) = E_t(t_i)\right\}$.

Les matrices d'entrée I^+ et de sortie O^+ ont une structure remarquable qui permet de les générer automatiquement :

$$I^+ = \begin{bmatrix} 1 & \cdots & 1 \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \qquad O^+ = \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ 1 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

La matrice I^+ représente le fait que la place p^+_1 est en amont (ou une entrée) de toutes les transitions du modèle. Cette place p^+_1 correspond au marquage M^+_0 qui est défini par $M^+_0(p^+_1) = 1$ et $M^+_0(p^+_i) = 0,\ \forall p_i \neq p^+_1$. M^+_0 est associé au label Ok.

La matrice O^+ représente le fait que toutes les places sont en aval (ou des sorties) d'une unique transition t_i hormis p^+_1 qui n'est en sortie d'aucune transition. Ces places p^+_i correspondent aux marquages accessibles à partir de M^+_0 qui sont définis par $M^+(p^+_i) = 1$ et $M^+(p^+_j) = 0,\ \forall p^+_j \neq p^+_i$. Ces marquages sont associés à un unique label $F^+_{t_i}$.

Le réseau de Petri obtenu est un réseau de Petri dit conservatif. Cela signifie que pour tout marquage accessible à partir de M^+_0, nous avons $\sum_{p^+_i} M(p^+_i) = constante$. Ici, la constante vaut un. À chaque marquage est donc associée une seule place. Dans la représentation de la figure 7.3, les places sont donc associées aux labels des marquages qui leur correspondent.

Le passage d'une hypothèse Ok vers des hypothèses de fautes $F^+_{t_i}$ est représenté par le tir d'une transition t_i. Le retour vers une hypothèse de bon comportement est alors impossible.

Pour le réseau de Petri $R^- =< P^-, T^-, I^-, O^-, M^-_0 >$:
- $P^- = \left\{p^-_i, i \in [1, \ldots, n^t_{BF} + 1]\right\}$;
- M^-_0 est défini par $M^-_0(p^-_1) = 1$ et $M^-_0(p^-_i) = 0, i \in [2, n^t_{BF} + 1]$ tel que $l(M^-_0) = Ok$;
- $T^- = \left\{t^-_i, i \in [1, \ldots, n^t_{BF}] | E_t(t^-_i) = e_{no}\right\}$.

Les matrices d'entrée I^- et de sortie O^- sont identiques aux matrices I^+ et O^+.

L'insertion d'un événement e_i associé à la transition t_i est représentée dans R^+ par une transition t_i^+. De la même manière, l'absence d'un événement e_i associé à la transition t_i est représentée dans R^- par une transition t_i^-. $F_{t_i}^+$ correspond à l'hypothèse que l'événement e_i associé à la transition t_i s'est inséré. $F_{t_i}^-$ correspond à l'hypothèse que l'événement e_i associé à la transition t_i est absent.

Remarque : les modèles de classes de faute sont construits de la même manière que pour les automates. Ainsi, les événements associés à t_i^+ et t_i sont les mêmes. L'événement associé à t_i^- correspond à un événement non observable. Les modèles peuvent également être modifiés en ajoutant des transitions à la suite des places $F_{t_i^+}$.

7.1.2 Exemple

Pour notre exemple, nous allons nous intéresser au réseau de Petri, noté $R_{BF} =< P_{BF}, T_{BF}, I_{BF}, O_{BF}, M_{0-BF} >$, représenté sur la figure 7.4. Ce réseau de Petri est défini par :

$$P_{BF} = \left\{p_1^{bf}, p_2^{bf}, p_3^{bf}, p_4^{bf}, p_5^{bf}\right\},\ M_{0-BF} = \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix},\ T_{BF} = \{t_1, t_2, t_3, t_4\}$$

Et les matrices d'entrée et de sortie :

$$I_{BF} = \begin{array}{c} \\ p_1^{bf} \\ p_2^{bf} \\ p_3^{bf} \\ p_4^{bf} \\ p_5^{bf} \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad O_{BF} = \begin{array}{c} \\ p_1^{bf} \\ p_2^{bf} \\ p_3^{bf} \\ p_4^{bf} \\ p_5^{bf} \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Pour ce réseau de Petri, des labels sont associés aux différents marquages accessibles à partir de M_{0-BF} tels que :

$$l(M_{0-BF}) = l(\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}) = p_1^{bf},\ l(M_1) = l(\begin{bmatrix}0\\1\\1\\0\\0\end{bmatrix}) = p_2^{bf}p_3^{bf},\ l(M_2) = l(\begin{bmatrix}0\\1\\0\\1\\0\end{bmatrix}) = p_2^{bf}p_4^{bf},$$

$$l(M_3) = l(\begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix}) = p_5^{bf}$$

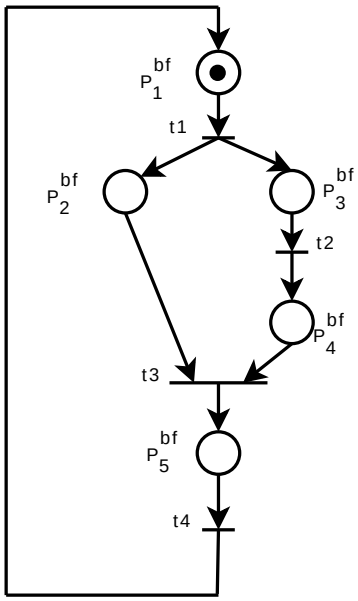


Figure 7.4 — Modèle de bon comportement R_{BF}

Les modèles de classe de fautes sont construits à partir de ce réseau de Petri. Nous obtenons pour le réseau de Petri $R^+ =< P^+, T^+, I^+, O^+, M_0^+ >$:

$$P^+ = \left\{p_1^+, p_2^+, p_3^+, p_4^+, p_5^+\right\},\ M_0^+ = \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix},\ T^+ = \left\{t_1^+, t_2^+, t_3^+, t_4^+\right\}$$

Et les matrices d'entrée et de sortie :

$$I^+ = \begin{array}{c} \\ p_1^+ \\ p_2^+ \\ p_3^+ \\ p_4^+ \\ p_5^+ \end{array} \begin{array}{c} \begin{array}{cccc} t_1^+ & t_2^+ & t_3^+ & t_4^+ \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad O^+ = \begin{array}{c} \\ p_1^+ \\ p_2^+ \\ p_3^+ \\ p_4^+ \\ p_5^+ \end{array} \begin{array}{c} \begin{array}{cccc} t_1^+ & t_2^+ & t_3^+ & t_4^+ \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Ici, les marquages correspondent aux places du réseau de Petri. Aussi, à chaque marquage, autrement dit, à chaque place, est associé un label :

$$l(M_1) = l(\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}) = Ok,\ l(M_2) = l(\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}) = F_{t_1}^+,\ l(M_3) = l(\begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix}) = F_{t_2}^+,\ l(M_4) = l(\begin{bmatrix}0\\0\\0\\1\\0\end{bmatrix}) = F_{t_3}^+,$$

$$l(M_5) = l(\begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix}) = F^+_{t_4}.$$

Le réseau de Petri $R^- =< P^-, T^-, I^-, O^-, M_0^- >$ est construit de la même manière :

$$P^- = \left\{p_1^-, p_2^-, p_3^-, p_4^-, p_5^-\right\},\ M_0^- = \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix},\ T^- = \left\{t_1^-, t_2^-, t_3^-, t_4^-\right\}$$

Les matrices de sortie et d'entrée, et les labels sont les mêmes que pour R^+. Il suffit de remplacer le signe + par le signe -. Les réseaux de Petri obtenus sont représentés sur les figures 7.5 et 7.6.

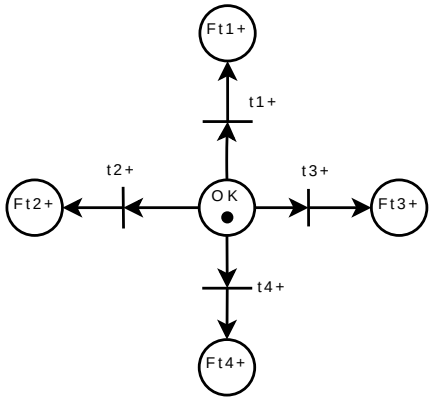


Figure 7.5 — Classe de faute R^+ : insertion

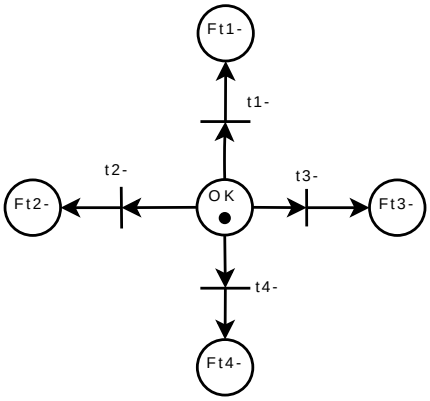


Figure 7.6 — Classe de faute R^- : absence

7.2 Modèles incluant des défaillances génériques

Comme avec les automates, le modèle de bon comportement est modifié en considérant l'insertion ou l'absence d'événement.

7.2.1 Modèle pour l'absence d'événement

L'absence d'un événement est représentée par l'ajout dans le modèle de bon comportement de transitions dont les événements associés sont non observables.

Nous supposons que seuls les événements attendus peuvent être absents. Ainsi, lorsque le modèle se trouve dans un marquage, il se peut que l'absence d'un événement ait eu lieu et, par conséquent, si le modèle se trouve dans l'état courant ou dans un marquage successeur. Pour représenter cette ambiguïté, des transitions t_i^- sont ajoutées pour chaque transition t_i

et dont les événements associés sont non observables. Les places d'entrée et de sortie sont identiques aux transitions t_i. Ainsi, lorsque le modèle se trouve dans un marquage, soit il n'y a pas d'absence d'événement et le modèle reste dans son marquage courant, soit il y a un événement qui est absent et par conséquent la transition associée à cet événement doit être tirée pour faire évoluer le modèle. Il y existe une ambiguïté sur le tir ou non de la transition t_i^-.

Ainsi,

$$\forall t_i \in T_{BF}, \exists t_i^- | I(.,t_i^-) = I(.,t_i),\ O(.,t_i^-) = O(.,t_i^-),\ \text{et}\ E_t(t_i^-) = e_{no}.$$

Le nouveau réseau de Petri, noté $R_{BF}^- =< P_{BF}^-, T_{BF}^-, I_{BF}^-, O_{BF}^-, M_{0-BF}^- >$ est défini par :
- $P_{BF}^- = P_{BF}$;
- $M_{0-BF}^- = M_{0-BF}$;
- $T_{BF}^- = T_{BF} \bigcup T^-$.

L'ensemble des places correspondent aux places du modèle de bon comportement. Nous faisons l'union des transitions du modèle de bon comportement avec celles de R^-. Le marquage initial correspond au marquage initial du modèle de bon comportement.

Les matrices d'entrée I_{BF}^- et de sortie O_{BF}^- sont définies par :

$$I_{BF}^- = \begin{array}{c} \\ p_1^{bf-} \\ \vdots \\ p_{n_{BF}^s}^{bf-} \end{array} \begin{array}{c} \begin{array}{ccc|ccc} t_1 & \cdots & t_{n_{BF}^t} & t_1^- & \cdots & t_{n_{BF}^t}^- \end{array} \\ \left[\begin{array}{ccc|ccc} & & & & & \\ & I_{BF} & & & I_{BF} & \\ & & & & & \end{array}\right] \end{array}$$

$$O_{BF}^- = \begin{array}{c} \\ p_1^{bf-} \\ \vdots \\ p_{n_{BF}^s}^{bf-} \end{array} \begin{array}{c} \begin{array}{ccc|ccc} t_1 & \cdots & t_{n_{BF}^t} & t_1^- & \cdots & t_{n_{BF}^t}^- \end{array} \\ \left[\begin{array}{ccc|ccc} & & & & & \\ & O_{BF} & & & O_{BF} & \\ & & & & & \end{array}\right] \end{array}$$

Les matrices d'entrée I_{BF}^- et de sortie O_{BF}^- indiquent que les transitions t_i^- ont les mêmes places d'entrée et de sortie que les transitions t_i. Il y a bien une duplication des transitions.

7.2.2 Modèle pour l'insertion d'événement

Représenter l'insertion d'un événement consiste à ajouter au modèle des transitions ne provoquant pas de changement de marquage.

Nous considérons seulement les événements qui peuvent faire changer le modèle de trajectoire c'est-à-dire les événements attendus par le marquage courant. Nous pouvons prendre en compte l'insertion de tous les événements à chaque marquage mais cela complexifie le modèle sans apporter réellement d'information supplémentaire. Cependant, lors d'une détection, l'événement incohérent est suspecté.

Pour représenter cette ambiguïté lors de la réception d'un événement (appartient-il au bon comportement ou est-il le produit d'une défaillance ?), des transitions notées t_i^+ sont ajoutées, avec des événements associés identiques à ceux des t_i, et dont la place d'entrée est la même que celle des t_i (transition neutre). Ainsi, lors de la réception de l'événement associé à t_i et t_i^+, une ambiguïté existe sur la transition à tirer. Soit l'événement reçu est issu du bon comportement et donc la transition t_i est tirée, soit l'événement reçu est le produit d'une défaillance et donc c'est la transition t_i^+ qui doit être tiré.

Lorsque la transition t_i^+ est tirée, alors nous devons rester dans le même état. Par conséquent, les transitions t_i^+ ont des places de sorties identiques à leurs places d'entrée.

Ainsi,

$$\forall t_i \in T_{BF}, \exists t_i^+ | I(.,t_i^+) = I(.,t_i),\ O(.,t_i^+) = I(.,t_i^+), \text{ et } E_t(t_i^+) = E_t(t_i).$$

Le réseau de Petri résultant, noté $R_{BF}^+ =< P_{BF}^+, T_{BF}^+, I_{BF}^+, O_{BF}^+, M_{0-BF}^+ >$ est défini par :
- $P_{BF}^+ = P_{BF}$;
- $M_{0-BF}^+ = M_{0-BF}$;
- $T_{BF}^+ = T_{BF} \bigcup T^+$.

L'ensemble des places correspondent aux places du modèle de bon comportement. Nous faisons l'union des transitions du modèle de bon comportement avec celles de R^+. Le marquage initial correspond au marquage initial du modèle de bon comportement.

Les matrices d'entrée I_{BF}^+ et de sortie O_{BF}^+ sont définies par :

$$I_{BF}^+ = \begin{array}{c} \\ p_1^{bf+} \\ \vdots \\ p_{n_{BF}^s}^{bf+} \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & \cdots & t_{n_{BF}^t} & t_1^+ & \cdots & t_{n_{BF}^t}^+ \end{array} \\ \left[\begin{array}{ccc|ccc} & & & & & \\ & I_{BF} & & & I_{BF} & \\ & & & & & \end{array}\right] \end{array}$$

$$O_{BF}^+ = \begin{array}{c} \\ p_1^{bf+} \\ \vdots \\ p_{n_{BF}^s}^{bf+} \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & \cdots & t_{n_{BF}^t} & t_1^+ & \cdots & t_{n_{BF}^t}^+ \end{array} \\ \left[\begin{array}{ccc|ccc} & & & & & \\ & O_{BF} & & & I_{BF} & \\ & & & & & \end{array}\right] \end{array}$$

La matrice d'entrée I_{BF}^+ indique que les transitions t_i^+ ont les mêmes places d'entrée que les transitions t_i. Par contre, la matrice de sortie O_{BF}^+ indique que les places de sortie des transitions t_i^+ correspondent aux places d'entrée des transitions t_i (et donc que t_i^+). Nous construisons bien des transitions ayant comme places de sortie leur place d'entrée.

Remarque : les deux réseaux de Petri ont autant de places que le modèle de bon comportement (qui sont $n_{BF+}^s = n_{BF}^s$ et $n_{BF-}^s = n_{BF}^s$) et deux fois plus de transitions (noté $n_{BF+}^t = 2 * n_{BF}^t$ et $n_{BF-}^t = 2 * n_{BF}^t$).

7.2.3 Exemple

Reprenons l'exemple précédent. Le nouveau réseau de Petri, noté $R^-_{BF} =< P^-_{BF}, T^-_{BF}, I^-_{BF}, O^-_{BF}, M^-_{0-BF} >$ est défini par :

$$P^-_{BF} = \left\{p_1^{bf-}, p_2^{bf-}, p_3^{bf-}, p_4^{bf-}, p_5^{bf-}\right\},\ M^-_{0-BF} = M_{0-BF} = \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix},$$

$$T^-_{BF} = T_{BF} \bigcup T^- = \left\{t_1, t_2, t_3, t_4, t_1^-, t_2^-, t_3^-, t_4^-\right\}.$$

Les matrices d'entrée I^-_{BF} et de sortie O^-_{BF} sont définies par :

$$I^-_{BF} = \begin{array}{c} \\ p_1^{bf-} \\ p_2^{bf-} \\ p_3^{bf-} \\ p_4^{bf-} \\ p_5^{bf-} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_1^- & t_2^- & t_3^- & t_4^- \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \quad O^-_{BF} = \begin{array}{c} \\ p_1^{bf-} \\ p_2^{bf-} \\ p_3^{bf-} \\ p_4^{bf-} \\ p_5^{bf-} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_1^- & t_2^- & t_3^- & t_4^- \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

De la même manière, le nouveau réseau de Petri, noté $R^+_{BF} =< P^+_{BF}, T^+_{BF}, I^+_{BF}, O^+_{BF}, M^+_{0-BF} >$ est défini par :

$$P^+_{BF} = \left\{p_1^{bf+}, p_2^{bf+}, p_3^{bf+}, p_4^{bf+}, p_5^{bf+}\right\},\ M^+_{0-BF} = M_{0-BF} = \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix},$$

$$T^+_{BF} = T_{BF} \bigcup T^+ = \left\{t_1, t_2, t_3, t_4, t_1^+, t_2^+, t_3^+, t_4^+\right\}.$$

Les matrices d'entrée I^+_{BF} et de sortie O^+_{BF} sont définies par :

$$I^+_{BF} = \begin{array}{c} \\ p_1^{bf+} \\ p_2^{bf+} \\ p_3^{bf+} \\ p_4^{bf+} \\ p_5^{bf+} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_1^+ & t_2^+ & t_3^+ & t_4^+ \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \quad O^+_{BF} = \begin{array}{c} \\ p_1^{bf+} \\ p_2^{bf+} \\ p_3^{bf+} \\ p_4^{bf+} \\ p_5^{bf+} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_1^+ & t_2^+ & t_3^+ & t_4^+ \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Les labels restent inchangés puisque les marquages accessibles restent les mêmes. De plus, nous avons $E_t(t_i^+) = E_t(t_i), i \in [1 \dots 4]$ et $E_t(t_i^-) = e_{no}, i \in [1 \dots 4]$.

Les réseaux de Petri obtenus sont représentés sur les figures 7.7 et 7.8.

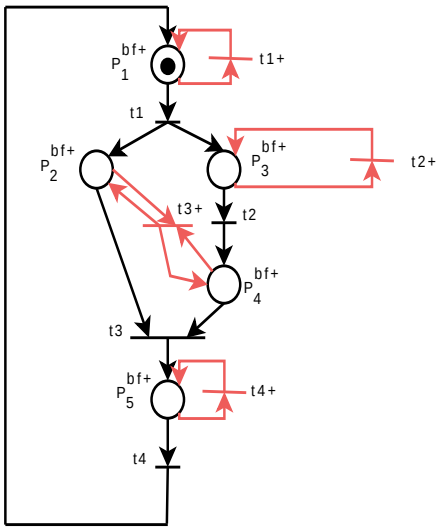


Figure 7.7 — RdP BF modifié R^{+}_{BF}

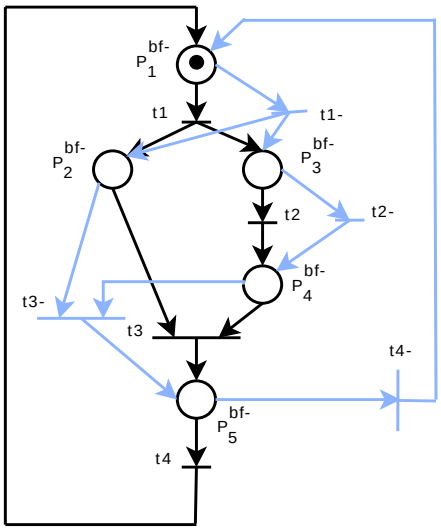


Figure 7.8 — RdP BF modifié R^{-}_{BF}

7.3 Synchronisation des modèles

7.3.1 La synchronisation des modèles

Comme il a été vu dans l'introduction, le formalisme des réseaux de Petri permet de synchroniser les modèles en fusionnant les transitions. Le modèle synchronisé comporte un nombre de places équivalent à la somme des places des modèles à synchroniser. L'utilisation de ce formalisme évite l'explosion combinatoire retrouvée dans le produit synchronisé des automates, qui lui nécessite la construction de l'ensemble des états accessibles (un produit d'états).

Après avoir construit ces différents modèles (i.e. $R^{+/-}_{BF}$ et $R^{+/-}$), une synchronisation est faite entre ces différents modèles afin d'associer les marquages du modèle de bon comportement avec ceux des classes de fautes. Ainsi, à chaque état de la fonction est associée une hypothèse, celle de bon comportement (OK) ou de faute ($F^{+}_{t_i}$ ou $F^{-}_{t_i}$).

Le réseau de Petri synchronisé pour l'absence d'événement, noté $R^{-}_{SYNC} =< P^{-}_{SYNC}, T^{-}_{SYNC}, I^{-}_{SYNC}, O^{-}_{SYNC}, M^{-}_{0-SYNC} >$ est défini par :

- $P^{-}_{SYNC} = P^{-}_{BF} \bigcup P^{-}$;
- $M^{-}_{0-SYNC} = \begin{bmatrix} M^{-}_{0-BF} \\ M^{-}_{0} \end{bmatrix}$;
- $T^{-}_{SYNC} = T^{-}_{BF}$.

Nous faisons l'union des places des deux réseaux R^{-}_{BF} et R^{-}. Le marquage initial correspond aux marquages initiaux des deux réseaux. Notons que $T^{-} \subseteq T^{-}_{BF}$.

Les matrices d'entrée I^{-}_{SYNC} et de sortie O^{-}_{SYNC} sont définies par :

$$I^-_{SYNC} = \begin{array}{c} \\ p_1^{sync-} \\ \vdots \\ p_{n^s_{BF}}^{sync-} \\ p_{n^s_{BF}+1}^{sync-} \\ \vdots \\ p_{n^s_{BF}+n^s_{ABS}}^{sync-} \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & \cdots & t_{n^t_{BF}} & t_1^- & \cdots & t_{n^t_{BF}}^- \end{array} \\ \left[\begin{array}{ccc|ccc} & & & & & \\ & I_{BF} & & & I_{BF} & \\ & & & & & \\ \hline & & & & & \\ & 0 & & & I^- & \\ & & & & & \end{array}\right] \end{array}$$

$$O^-_{SYNC} = \begin{array}{c} \\ p_1^{sync-} \\ \vdots \\ p_{n^s_{BF}}^{sync-} \\ p_{n^s_{BF}+1}^{sync-} \\ \vdots \\ p_{n^s_{BF}+n^s_{ABS}}^{sync-} \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & \cdots & t_{n^t_{BF}} & t_1^- & \cdots & t_{n^t_{BF}}^- \end{array} \\ \left[\begin{array}{ccc|ccc} & & & & & \\ & O_{BF} & & & O_{BF} & \\ & & & & & \\ \hline & & & & & \\ & 0 & & & O^- & \\ & & & & & \end{array}\right] \end{array}$$

La synchronisation des modèles se fait sur les transitions t_i^-. Ces transitions t_i^- ont donc comme places d'entrée et de sortie celles qu'elles avaient dans le réseau de Petri R^-_{BF} et celles qu'elles avaient dans le réseau de Petri R^-. Il n'y a pas de synchronisation sur les transitions t_i donc les places amont et aval des t_i sont identiques à celles des t_i dans R^-_{BF}.

Le réseau de Petri synchronisé pour l'insertion d'événement, noté $R^+_{SYNC} =< P^+_{SYNC}$,- $T^+_{SYNC}, I^+_{SYNC}, O^+_{SYNC}, M^+_{0-SYNC} >$ est défini par :

- $P^+_{SYNC} = P^+_{BF} \bigcup P^+$;
- $M^+_{0-SYNC} = \begin{bmatrix} M^+_{0-BF} \\ M^+_0 \end{bmatrix}$;
- $T^+_{SYNC} = T^+_{BF}$.

Nous faisons l'union des places des deux réseaux R^+_{BF} et R^+. Le marquage initial correspond aux marquages initiaux des deux réseaux. Notons que $T^+ \subseteq T^+_{BF}$.

Les matrices d'entrée I^+_{SYNC} et de sortie O^+_{SYNC} sont définies par :

$$I^+_{SYNC} = \begin{array}{c} \\ p_1^{sync+} \\ \vdots \\ p_{n^s_{BF}}^{sync+} \\ p_{n^s_{BF}+1}^{sync+} \\ \vdots \\ p_{n^s_{BF}+n^s_{ABS}}^{sync+} \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & \cdots & t_{n^t_{BF}} & t_1^+ & \cdots & t_{n^t_{BF}}^+ \end{array} \\ \left[\begin{array}{ccc|ccc} & & & & & \\ & I_{BF} & & & I_{BF} & \\ & & & & & \\ \hline & & & & & \\ & 0 & & & I^+ & \\ & & & & & \end{array}\right] \end{array}$$

$$O^+_{SYNC} = \begin{array}{c} \\ p_1^{sync+} \\ \vdots \\ p_{n^s_{BF}}^{sync+} \\ p_{n^s_{BF}+1}^{sync+} \\ \vdots \\ p_{n^s_{BF}+n^s_{ABS}}^{sync+} \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & \cdots & t_{n^t_{BF}} & t^+_1 & \cdots & t^+_{n^t_{BF}} \end{array} \\ \left[\begin{array}{ccc|ccc} & & & & & \\ & O_{BF} & & & I_{BF} & \\ & & & & & \\ \hline & & & & & \\ & 0 & & & O^+ & \\ & & & & & \end{array}\right] \end{array}$$

Revenons sur la notion de label d'un marquage. Un label est associé à chaque marquage accessible à partir du marquage initial. Ici, le marquage est décomposé de la manière suivante :

$$M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$$

avec M_1 appartenant à l'ensemble des marquages accessibles à partir de M^+_{0-BF} de R^+_{BF} et M_2 appartenant à l'ensemble des marquages accessibles à partir de M^+_0 de R^+.

En effet, $M^+_{0-SYNC} = [M^+_{0-BF}; M^+_0]$ avec M^+_{0-BF} appartenant à l'ensemble des marquages accessibles à partir de M^+_{0-BF} de R^+_{BF} et M^+_0 appartenant à l'ensemble des marquages accessibles à partir de M^+_0 de R^+.

Supposons que cela est vrai pour un marquage M accessible de R^+_{SYNC} à partir de M^+_{0-SYNC}. Soit t une transition sensibilisée par M i.e. telle que :

$$M \geq I^+_{SYNC}(.,t)$$

Le marquage atteint par franchissement de t est défini soit par :

$$\begin{aligned} M' &= M + O^+_{SYNC}(.,t) - I^+_{SYNC}(.,t) = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} + \begin{bmatrix} I_{BF}(.,t) \\ O^+(.,t) \end{bmatrix} - \begin{bmatrix} I_{BF}(.,t) \\ I^+(.,t) \end{bmatrix} \\ &= \begin{bmatrix} M_1 + I_{BF}(.,t) - I_{BF}(.,t) \\ M_2 + O^+(.,t) - I^+(.,t) \end{bmatrix} = \begin{bmatrix} M_1 \\ M'_2 \end{bmatrix} \end{aligned}$$

soit par :

$$\begin{aligned} M' &= M + O^+_{SYNC}(.,t) - I^+_{SYNC}(.,t) = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} + \begin{bmatrix} O_{BF}(.,t) \\ 0 \end{bmatrix} - \begin{bmatrix} I_{BF}(.,t) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} M_1 + O_{BF}(.,t) - I_{BF}(.,t) \\ M_2 + 0 \end{bmatrix} = \begin{bmatrix} M'_1 \\ M_2 \end{bmatrix} \end{aligned}$$

Dans les deux cas, les marquages M'_1 et M'_2 obtenus appartiennent respectivement à l'ensemble des marquages accessibles de R^+_{BF} et de R^+ à partir de M^+_{0-BF} et M^+_0.

Nous avons donc pour tout marquage accessible M à partir de M^+_{0-SYNC} une décomposition de M en marquages M_1 de R^+_{BF} et M_2 de R^+.

Nous associons alors à M un label de la manière suivante :

$$l(M) = (l(M_1), l(M_2))$$

7.3.2 Exemple

Construisons maintenant les réseaux de Petri synchronisés. Le réseau de Petri synchronisé pour l'absence d'événement, noté $R^-_{SYNC} =< P^-_{SYNC}, T^-_{SYNC}, I^-_{SYNC}, O^-_{SYNC}, M^-_{0-SYNC} >$ est défini par :

$$\begin{aligned} P^-_{SYNC} &= P^-_{BF} \bigcup P^- = \left\{p_1^{bf-}, p_2^{bf-}, p_3^{bf-}, p_4^{bf-}, p_5^{bf-}, p_1^-, p_2^-, p_3^-, p_4^-, p_5^-\right\} \\ &= \left\{p_1^{sync-}, p_2^{sync-}, p_3^{sync-}, p_4^{sync-}, p_5^{sync-}, p_6^{sync-}, p_7^{sync-}, p_8^{sync-}, p_9^{sync-}, p_{10}^{sync-}\right\} \end{aligned}$$

$$\begin{aligned} M^-_{0-SYNC} &= [M^-_{0-BF}; M^-_0] = [1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T; \\ T^-_{SYNC} &= T_{BF} \bigcup T^- = \left\{t_1, t_2, t_3, t_4, t_1^-, t_2^-, t_3^-, t_4^-\right\} \end{aligned}$$

Les matrices d'entrée I^-_{SYNC} et de sortie O^-_{SYNC} sont définies par :

$$I^-_{SYNC} = \begin{array}{c} \\ p_1^{sync-} \\ p_2^{sync-} \\ p_3^{sync-} \\ p_4^{sync-} \\ p_5^{sync-} \\ p_6^{sync-} \\ p_7^{sync-} \\ p_8^{sync-} \\ p_9^{sync-} \\ p_{10}^{sync-} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_1^- & t_2^- & t_3^- & t_4^- \end{array} \\ \left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \quad O^-_{SYNC} = \begin{array}{c} \\ p_1^{sync-} \\ p_2^{sync-} \\ p_3^{sync-} \\ p_4^{sync-} \\ p_5^{sync-} \\ p_6^{sync-} \\ p_7^{sync-} \\ p_8^{sync-} \\ p_9^{sync-} \\ p_{10}^{sync-} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_1^- & t_2^- & t_3^- & t_4^- \end{array} \\ \left[\begin{array}{cccccccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{array}$$

Nous pouvons également associer des labels aux différents marquages accessibles :

$$\begin{aligned} l(M^+_{0-SYNC}) &= l([1000010000]^T) = (1, Ok) \\ l(M_1) &= l([0110001000]^T) = (2, 3, F^-_{t_1}) \\ l(M_2) &= l([0110010000]^T) = (2, 3, Ok) \\ l(M_3) &= l([0101000100]^T) = (2, 4, F^-_{t_2}) \\ l(M_4) &= l([0101010000]^T) = (2, 4, Ok) \\ \vdots &\ \vdots\ \ \vdots\ \ \vdots\ \ \vdots \\ l(M_k) &= l([0000110000]^T) = (5, Ok) \\ l(M_{k+1}) &= l([1000000001]^T) = (1, F^-_{t_4}) \end{aligned}$$

Le réseau de Petri synchronisé pour l'insertion d'événement, noté $R^+_{SYNC} =< P^+_{SYNC}$,- $T^+_{SYNC}, I^+_{SYNC}, O^+_{SYNC}, M^+_{0-SYNC} >$ est défini par :

$$\begin{aligned} P^+_{SYNC} &= P^+_{BF} \bigcup P^+ = \left\{p_1^{bf+}, p_2^{bf+}, p_3^{bf+}, p_4^{bf+}, p_5^{bf+}, p_1^+, p_2^+, p_3^+, p_4^+, p_5^+\right\} \\ &= \left\{p_1^{sync+}, p_2^{sync+}, p_3^{sync+}, p_4^{sync+}, p_5^{sync+}, p_6^{sync+}, p_7^{sync+}, p_8^{sync+}, p_9^{sync+}, p_{10}^{sync+}\right\} \end{aligned}$$

$$\begin{aligned} M^+_{0-SYNC} &= [M^+_{0-BF}; M^+_0] = [1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T; \\ T^+_{SYNC} &= T_{BF} \bigcup T^+ = \left\{t_1, t_2, t_3, t_4, t_1^+, t_2^+, t_3^+, t_4^+\right\}. \end{aligned}$$

Les matrices d'entrée I^+_{SYNC} et de sortie O^+_{SYNC} sont définies par :

$$I^+_{SYNC} = \begin{array}{c} \\ p_1^{sync+} \\ p_2^{sync+} \\ p_3^{sync+} \\ p_4^{sync+} \\ p_5^{sync+} \\ p_6^{sync+} \\ p_7^{sync+} \\ p_8^{sync+} \\ p_9^{sync+} \\ p_{10}^{sync+} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_1^+ & t_2^+ & t_3^+ & t_4^+ \end{array} \\ \left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

$$O^+_{SYNC} = \begin{array}{c} \\ p_1^{sync+} \\ p_2^{sync+} \\ p_3^{sync+} \\ p_4^{sync+} \\ p_5^{sync+} \\ p_6^{sync+} \\ p_7^{sync+} \\ p_8^{sync+} \\ p_9^{sync+} \\ p_{10}^{sync+} \end{array} \begin{array}{c} \begin{array}{cccccccc} t_1 & t_2 & t_3 & t_4 & t_1^+ & t_2^+ & t_3^+ & t_4^+ \end{array} \\ \left[\begin{array}{cccccccc} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{array}$$

Nous pouvons également associer des labels aux différents marquages accessibles :

$$\begin{array}{lcll} l(M^+_{0-SYNC}) & = & l([1000010000]^T) & = & (1, Ok) \\ l(M_1) & = & l([0110010000]^T) & = & (2, 3, Ok) \\ l(M_2) & = & l([1000001000]^T) & = & (1, F^+_{t_1}) \\ l(M_3) & = & l([0110000100]^T) & = & (2, 3, F^+_{t_2}) \\ l(M_4) & = & l([0101010000]^T) & = & (2, 4, Ok) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ l(M_k) & = & l([0000110000]^T) & = & (5, Ok) \\ l(M_{k+1}) & = & l([0000100001]^T) & = & (5, F^+_{t_4}) \end{array}$$

Les représentations graphiques de ces réseaux de Petri sont montrées sur les figures 7.9 et 7.10.

7.4 Les Diagnostiqueurs

La synchronisation des modèles ne prend pas en compte le fait qu'un événement associé à t_i^+ est le même que celui associé à t_i et que l'événement associé à t_i^- est non observable. Pour résoudre ce problème, nous travaillons avec le graphe des marquages accessibles. Si ce graphe est construit dans son intégralité alors il est équivalent au produit synchronisé vu dans le chapitre précédent (chapitre 6.3). Néanmoins, l'information contenue dans les synchronisations des modèles permet d'utiliser et de modifier l'algorithme pour ne pas développer l'ensemble du graphe des marquages accessibles. En effet, comme pour la construction du diagnostiqueur automate, dès qu'un marquage contenant seulement des marquages associés à des labels de fautes est atteint, une détection est faite. Par conséquent, nous choisissons d'arrêter la construction du graphe.

L'avantage, par rapport, aux automates est que la synchronisation des modèles apporte de l'information qui va être utilisée dans la construction du graphe des marquages accessibles. Cela évite la construction de l'ensemble du graphe. Pour les automates, il faut d'abord

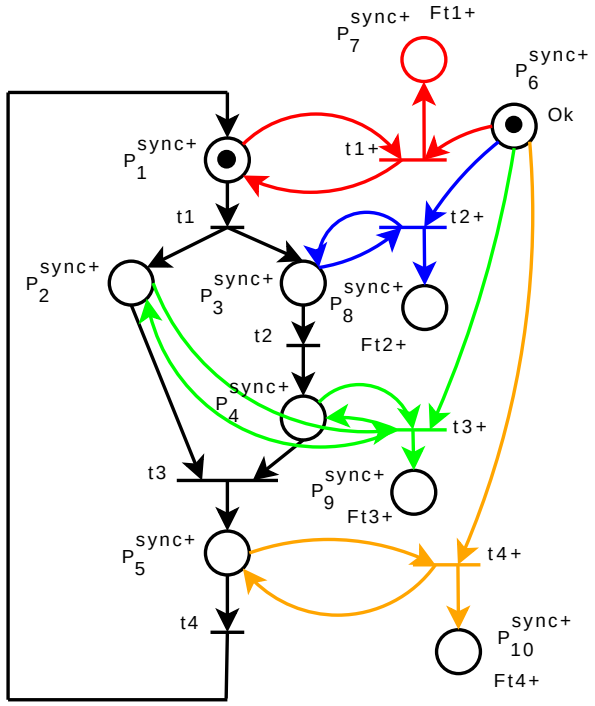


Figure 7.9 — RdP Synchronisé R^{+}_{SYNC} : insertion

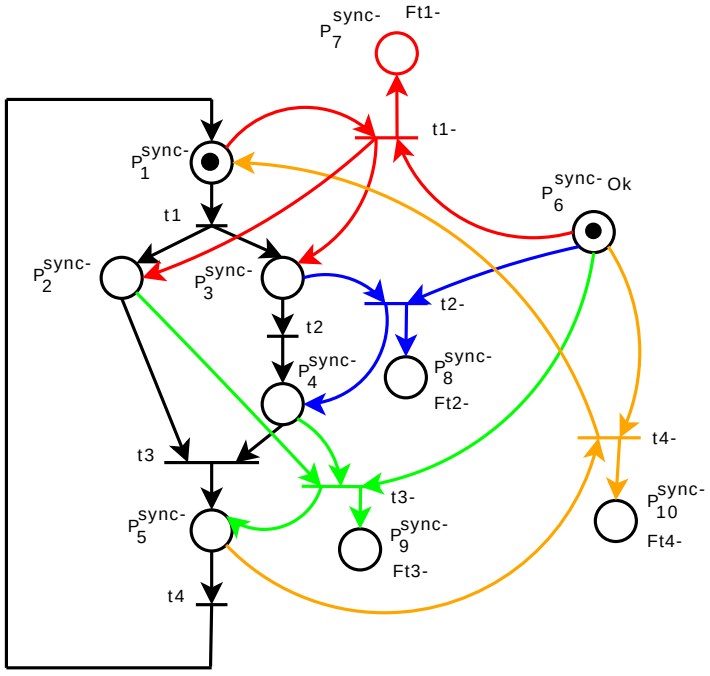


Figure 7.10 — RdP Synchronisé R^{-}_{SYNC} : absence

construire l'ensemble du produit synchronisé et ensuite le modifier pour obtenir un diagnostiqueur.

Nous pouvons également, dans le cadre d'une étude de diagnosticabilité, établir une condition d'arrêt dès qu'un marquage contenant un seul label de faute est atteint.

La construction du graphe des marquages accessibles permet de réunir les marquages associés aux places amonts et avals d'une transition non observable ou les marquages associés aux places avals des transitions ayant les mêmes événements associés et les mêmes places amonts.

7.4.1 Algorithme de Karp et Miller

Pour la construction du graphe d'accessibilité, nous allons utiliser une version simplifié de l'algorithme de Karp et Miller (Karp et Miller, 1969). Cet algorithme est représenté sur l'algorithme 7.1. En effet, le modèle de bon comportement et les modèles de classes de fautes sont bornés pour leur marquage initial c'est-à-dire que pour tous les marquages accessibles à partir du marquage initial, le nombre de jetons dans chacune des places du réseau de Petri est fini. Par conséquent, le modèle représentant la synchronisation des réseaux de Petri est également borné. Ainsi, l'espace d'états est fini et il n'y aura pas de feuille correspondant à une divergence du marquage. La condition de vérification de la divergence du marquage peut donc être supprimée de l'algorithme.

Une version simplifiée de l'algorithme de Karp et Miller est donc présenté avec l'algorithme 7.1.

Principe : il se construit progressivement à partir du marquage initial qui est associé à la racine de l'arbre. Un successeur M' de M est créé si et seulement si $\exists t_i \in T$ tq $M \geq I(, t_i)$.

Pour chaque noeud de l'arbre nouvellement créé, l'algorithme suivant est appliqué dont le but est d'obtenir une arborescence dans laquelle toutes les feuilles sont qualifiées.

Algorithme 7.1 — Algorithme simplifié de Karp et Miller

```
   Entrées : Réseau de Petri
   Sorties : Arbre de couverture
 1 pour Pour chaque feuille (Mf) de l'arbre non qualifiée faire
 2 |   si il existe un autre noeud associé au même marquage Mf DONT TOUS LES
   |   SUCCESSEURS ont été ou vont être calculés alors
 3 |   |   la feuille est qualifiée de DUPLICAT
 4 |   sinon
 5 |   |   si il n'existe aucune transition franchissable à partir de Mf alors
 6 |   |   |   la feuille est qualifiée de TERMINALE
 7 |   |   sinon
 8 |   |   |   construire tous les successeurs de la feuille
 9 |   |   fin
10 |   fin
11 fin
```

Dans le cas d'un réseau borné (le cas ici), le graphe des marquages accessibles est construit en fusionnant les feuilles "DUPLICAT" avec les nœuds correspondants.

Cet algorithme construit l'ensemble des marquages accessibles à partir d'un marquage initial. Nous choisissons de ne pas développer tous les marquages accessibles car nous voulons arrêter cette construction dès que le modèle sort du bon comportement et donc dès qu'il y

a une détection. Nous ne construisons donc pas l'ensemble des marquages accessibles mais une sous-partie de ce graphe. **Nous réduisons ainsi le nombre de marquages explorés à ce qui est nécessaire.** Voyons comment modifier cet algorithme pour construire les diagnostiqueurs.

7.4.2 Absence d'un événement

7.4.2.1 Principe

Dans le modèle synchronisé, il existe des transitions qui sont associées à des événements non observables. Il existe donc des marquages qui sont liés par ces transitions non observables. Il y a donc une ambiguïté sur les marquages. Pour représenter cette ambiguïté, les marquages liés par de telles transitions sont rassemblés dans un même ensemble. Nous ne raisonnons plus sur des marquages mais sur des ensembles de marquages. Nous appelons un ensemble $\mathcal{M}_f$, un ensemble de marquages M ($\mathcal{M}_f$=$\{M\}$). Les événements non observables représentent l'absence d'un événement et l'ensemble des marquages exprime l'ambiguïté sur les marquages (est-on dans un marquage avec un bon comportement, ou dans un autre marquage en présence d'une faute ?).

En modifiant ainsi les modèles, il faut vérifier que les ensembles de marquages n'ont pas de transitions de sortie dont les événements associés sont identiques. En effet, si c'est le cas, alors, lors de la reception de cet événement, il existera une ambiguïté représentée par un conflit pour les transitions. Pour représenter cette ambiguïté, il faut à nouveau rassembler les ensembles de marquages successeurs dans un même ensemble.

7.4.2.2 Formalisation

Finalement, l'ambiguïté sur les marquages liés par des transitions non observables se traduit par :

$$\text{soit un marquage M tel que } M \geq I(.,t_i^-) \text{ avec } E_t(t_i^-) = e_{no} \text{ et } M' = M + C(.,t_i^-),$$
$$\text{alors } \mathcal{M}_f=\{M, M'\}.$$

L'ambiguïté sur les marquages dont les transitions d'entrée sont associées au même événement se traduit par :

$$\text{soit un marquage M tel que } M \geq I(.,t_i) \text{ et } M \geq I(.,t_j) \text{ avec } E_t(t_i) = E_t(t_j), \text{ et}$$
$$M' = M + C(.,t_i) \text{ et } M'' = M + C(.,t_j),$$
$$\text{alors } \mathcal{M}_f=\{M', M''\}.$$

L'algorithme 7.2 représente l'algorithme permettant de construire le graphe d'accessibilité pour l'absence.

Algorithme 7.2 — Graphe d'accessibilité pour l'absence

Entrées : Réseau de Petri, lien transitions/événements (fonction E_t)
Sorties : Graphe d'accessibilité pour l'absence

$\mathcal{M}_f$=$\{M_{0-SYNC}\}$
pour *chaque ensemble non qualifié* $\mathcal{M}_f$ **faire**
  si *il existe un autre nœud associé au même ensemble* $\mathcal{M}_f$ *DONT TOUS LES SUCCESSEURS ont été ou vont être calculés* **alors**
    l'ensemble est qualifié de DUPLICAT
  sinon
    si *il n'existe aucun marquage* $M \subseteq \mathcal{M}_f$ *tel que* $l(M) = (., Ok)$ *ou il existe aucune transition franchissable* $\forall M \subseteq \mathcal{M}_f$ **alors**
      l'ensemble est qualifié de TERMINALE
    sinon
      si *il existe au moins un marquage* $M \subseteq \mathcal{M}_f$ *tel que une transition sensibilisée par ce marquage ait comme événement non observable associé* **alors**
        construire tous les successeurs à partir des marquages $M \subseteq \mathcal{M}_f$ pour chaque t_i telle que $E_t(t_i) = e_{no}$, i.e. les t_i^-
        faire l'union entre $\mathcal{M}_f$ et les successeurs calculés
      fin
      si *il existe un autre nœud associé au même ensemble* $\mathcal{M}_f$ *DONT TOUS LES SUCCESSEURS ont été ou vont être calculés* **alors**
        l'ensemble est qualifié de DUPLICAT
      sinon
        construire tous les successeurs à partir des marquages M du nouvel ensemble $\mathcal{M}_f$ sauf à partir de ceux ayant eu une transition d'entrée t_i telle que $E_t(t_i) = e_{no}$
        faire l'union de tous les successeurs dont les transitions d'entrée t_i et t_i' sont telles que $E_t(t_i) = E_t(t_i')$
      fin
    fin
  fin
fin

7.4.3 Insertion d'un événement

7.4.3.1 Principe

Dans le modèle synchronisé, il existe des marquages dont les transitions de sorties sont associés au même événement. Comme dans le cas de l'absence, il existe une ambiguïté sur les marquages, représentée par un conflit sur les transitions. Il n'y a pas de certitude sur le marquage successeur. Pour représenter cette ambiguïté, ceux-ci sont donc rassemblés dans un même ensemble.

De la même manière que pour le diagnostiqueur absence, il faut vérifier si les transitions

de sortie d'un même ensemble de marquages ne sont pas associées aux mêmes événements et rassembler les ensembles si c'est le cas.

7.4.3.2 Formalisation

L'ambiguïté sur les marquages dont les transitions d'entrée sont associées au même événement se traduit par :

soit un marquage M tel que $M \geq I(.,t_i)$ et $M \geq I(.,t_j)$ avec $E_t(t_i) = E_t(t_j)$, et
$M' = M + C(.,t_i)$ et $M'' = M + C(.,t_j)$,
alors $\mathcal{M}_f$=$\{M', M''\}$.

L'algorithme 7.3 représente l'algorithme permettant de construire le graphe d'accessibilité pour l'insertion.

Algorithme 7.3 — Graphe d'accessibilité pour l'insertion

Entrées : Réseau de Petri, lien transitions/événements (fonction E_t)
Sorties : Graphe d'accessibilité pour l'insertion

```
1  M_f = {M+_{0-SYNC}}
2  pour chaque ensemble non qualifié M_f faire
3  |  si il existe un autre nœud associé au même ensemble M_f DONT TOUS LES
   |  SUCCESSEURS ont été ou vont être calculés alors
4  |  |  l'ensemble est qualifié de DUPLICAT
5  |  sinon
6  |  |  si il n'existe aucun marquage M ⊆ M_f tel que l(M) = (.,Ok) ou il n'existe
   |  |  aucune transition franchissable ∀M ⊆ M_f alors
7  |  |  |  l'ensemble est qualifié de TERMINALE
8  |  |  sinon
9  |  |  |  construire tous les successeurs à partir des marquages M ⊆ M_f
10 |  |  |  faire l'union de tous les successeurs dont les transitions d'entrée t_i et t'_i sont
   |  |  |  telles que E_t(t_i) = E_t(t'_i)
11 |  |  fin
12 |  fin
13 fin
```

Pour des aspects applicatifs (voire le chapitre 8), les graphes obtenus sont ensuite représentés sous forme de réseaux de Petri. Ces modèles sont en réalité les diagnostiqueurs pour l'absence et l'insertion d'un événement. Ils correspondent aux diagnostiqueurs automates définis dans le chapitre précédent (Chapitre 6).

7.4.4 Exemple

En reprenant toujours notre exemple, les graphes d'accessibilité sont construits à partir des algorithmes définis ci-dessus (algorithmes 7.2 et 7.3).

Pour les deux algorithmes, le marquage initial est :

$$M^{+}_{0-SYNC} = M^{-}_{0-SYNC} = [1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T = [P_1, P_6]^T$$

7.4.4.1 Diagnostiqueur absence

À partir du marquage initial, une transition t_1^- sensibilisée est associée à un événement non observable. Cela correspond au fait que l'événement associé à t_1 puisse être absent. Aussi, soit le modèle se trouve dans l'état $[P1, P6]$, i.e. en $(1, Ok)$, soit dans l'état $[P2, P3, P7]$ i.e. en $(2, 3, F_{t_1^-})$. Ces deux marquages sont donc réunis.

Les marquages suivants sont construits à partir de ces deux marquages. Il s'agit de $[P2, P4, P7]$ i.e. $(2, 4, F_{t_1^-})$ et $[P2, P3, P6]$ i.e. $(2, 3, Ok)$. Ces deux marquages sont issus des transitions t_1 et t_2 sensibilisées à partir des marquages $[P2, P3, P6]$ et $[P2, P4, P7]$ respectivement avec que $E_t(t_1) \neq E_t(t_2)$. Les marquages ne sont donc pas réunis.

Considérons l'ensemble $\mathcal{M}_f = \{[P2, P4, P7]\}$. Un seul marquage M est inclus dans cet ensemble auquel est associé un label $l(M) = (2, 4, F_{t_1^-}) \neq (., Ok)$. Cet ensemble est qualifié de TERMINAL.

Le deuxième ensemble considéré est $\mathcal{M}_f = \{[P2, P3, P6]\}$. Cet ensemble contient un marquage M auquel est associé un label $l(M) = (2, 3, Ok)$. Il existe également une transition t_2^- sensibilisée à partir de $[P2, P3, P6]$ telle que $E_t(t_2^-) = e_{no}$. Le successeur de ce marquage est construit et les deux marquages sont rassemblés dans $\mathcal{M}_f$.

L'algorithme continue ainsi de suite jusqu'à obtenir l'ensemble $\mathcal{M}_f = \{[P1, P6], [P2, P3, P7]\}$ qui a déjà été calculé et qui est donc qualifié de DUPLICAT. Il ne reste plus d'ensemble à développer.

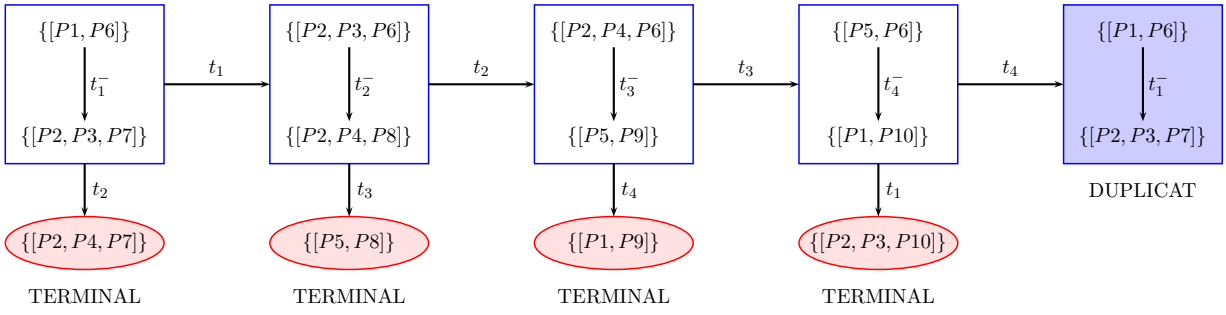


Figure 7.11 — Arbre de couverture pour le diagnostiqueur Absence

À partir de ce graphe, un réseau de Petri associé est construit et est représenté sur la figure 7.12. Chaque ensemble est représenté par une place labellisée par l'ensemble des labels

de l'ensemble. Les arcs définis entre les ensembles sont représentés par des transitions. Les transitions t_i^- sont supprimées du graphe des marquages et du réseau de Petri associé.

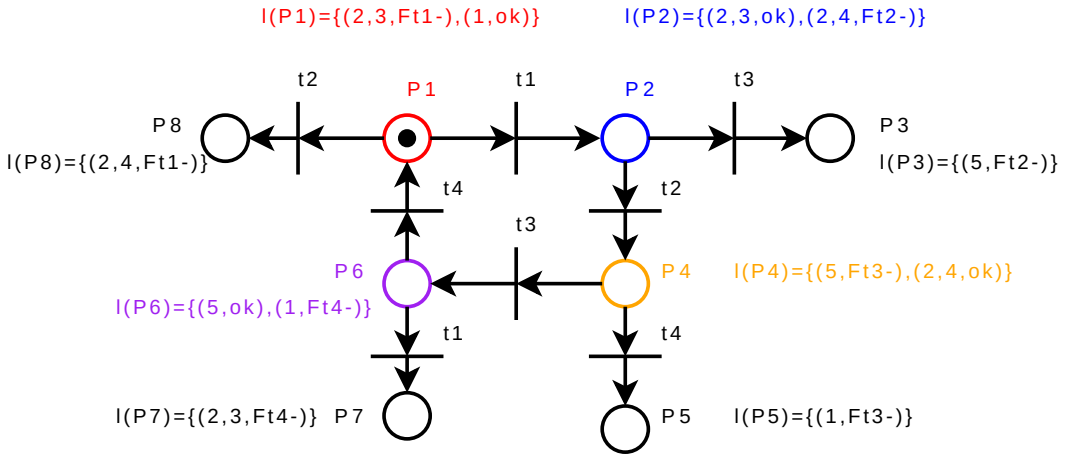


Figure 7.12 — RdP Diagnostiqueur : absence

7.4.4.2 Diagnostiqueur insertion

À partir du marquage initial, deux marquages $[P2, P3, P6]$ et $[P1, P7]$ sont construits à partir des transitions sensibilisées t_1 et t_1^+. Ces deux transitions sont telles que $E_t(t_1) = E_t(t_1^+)$. Les deux marquages obtenus sont donc réunis dans le même ensemble.

À partir de ce nouvel ensemble, les marquages suivants sont à nouveau construits. Deux d'entre eux proviennent des transitions sensibilisées t_2 et t_2^+ auxquelles sont associées un même événement. Ils sont donc rassemblés dans un même ensemble $\{[P2, P3, P8], [P2, P4, P6]\}$.

Un autre marquage $[P2, P3, P7]$ est obtenu et reste l'unique élément de son ensemble $\{[P2, P3, P7]\}$. Ce dernier ensemble n'a pas de label Ok associé ($l(M) = (2, 3, F_{t_1}^+) \neq (., Ok)$). Cet ensemble est qualifié de TERMINAL.

L'algorithme continue à construire le graphe de la même manière pour finalement atteindre le marquage $\{[P2, P3, P6], [P1, P7]\}$ qui a déjà était développé. Cet ensemble est donc qualifié de DUPLICAT. Il ne reste plus d'ensemble à développer.

À partir de ce graphe, un réseau de Petri associé est construit et est représenté sur la figure 7.14. Chaque ensemble est représenté par une place labellisée par l'ensemble des labels de l'ensemble. Les arcs définis entre les ensembles sont représentés par des transitions. Les transitions t_i^+ sont supprimées du graphe des marquages et du réseau de Petri associé.

7.5 Conclusion

Dans ce chapitre, une approche a été établie qui permet de localiser un événement qui s'est inséré ou qui est manquant. Cette approche utilise seulement les connaissances du système, à savoir son comportement normal.

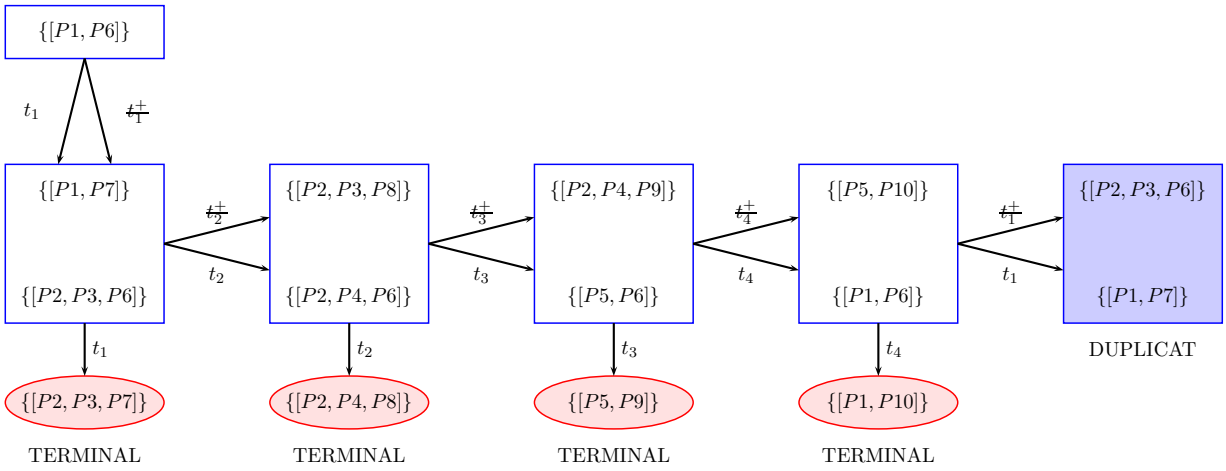


Figure 7.13 — Arbre de couverture pour le diagnostiqueur insertion

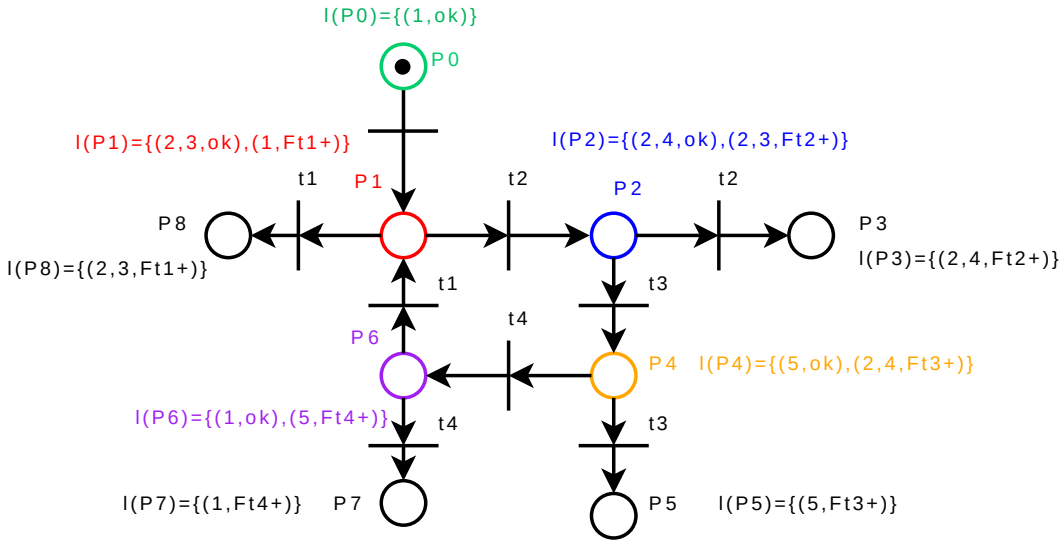


Figure 7.14 — RdP Diagnostiqueur : insertion

L'approche présentée résout le problème posé dans le chapitre précédent et qui concernait l'explosion du nombre d'états lors du produit synchronisé des automates. Ici, nous nous basons sur les informations contenues dans le modèle représentant la synchronisation des modèles pour pouvoir construire le diagnostiqueur. Nous conservons ainsi le principal avantage d'un diagnostiqueur à savoir la compilation de l'information de diagnostic dans un modèle. Comme précédemment, il suffit de suivre en ligne le diagnostiqueur et une fois qu'un événement est détecté, l'état courant du réseau de Petri permet d'émettre des hypothèses d'insertion ou d'absence d'un événement. Ceci permet de faire un minimum de calculs en ligne (juste un suivi en ligne).

La construction du diagnostiqueur se fait à partir du modèle à événements discrets représentant le bon comportement du système. Divers modèles sont construits de façon automa-

tique et sont ensuite utilisés pour obtenir au final le diagnostiqueur.

Dans le prochain chapitre, nous allons appliquer la méthode que nous avons présentée ici sur un système lié au domaine automobile.

8 Application de l'approche diagnostiqueur au domaine automobile

Introduction

Dans le chapitre 2, nous avons décrit l'architecture électronique d'un véhicule automobile.

Il y a des capteurs qui fournissent des informations aux calculateurs auxquels ils sont reliés. Ces calculateurs sont également reliés à des actionneurs auxquels ils donnent des ordres. Ces calculateurs sont connectés à un réseau de communication qui leur permet de s'échanger des informations. Ces informations sont utilisées par les algorithmes implémentés dans ces calculateurs afin de mettre en œuvre toutes les fonctions du véhicule.

Notre objectif est de suivre en ligne le comportement de telles fonctions en surveillant les messages échangés sur le réseau de communication, de détecter et localiser des défaillances intermittentes dans le système.

Les travaux présentés dans les chapitres précédents ont abouti à la mise en œuvre d'applications permettant de surveiller le réseau de communication, de suivre l'évolution du comportement d'une fonction et, de détecter et localiser des défaillances fugitives et intermittentes. Le logiciel Roméo et l'application DDP en sont les résultats.

Le développement de ces applications a permis de valider notre approche sur des cas concrets de défaillances intermittentes dans le domaine de l'automobile. Nous allons illustrer cette approche sur une fonction réelle, la fonction "gestion de la rampe d'accès pour handicapés", telle qu'elle est implémentée sur de nombreux transports collectifs urbains (bus, tramways, etc.).

8.1 Application

8.1.1 Le banc de test

Pour le domaine automobile, l'architecture multiplexée est devenue indispensable à cause de l'augmentation de l'électronique et par conséquent, du nombre de faisceaux électriques nécessaires. Ceci est encore plus vrai dans les autobus de ville car leur dimension oblige à

avoir de grandes longueurs de câbles.

La société Actia développe les différents calculateurs ainsi que les différents logiciels et programmes informatiques utilisés dans les autobus. L'architecture multiplexée utilisée par Actia dans les autobus est décrite sur la figure 8.1.

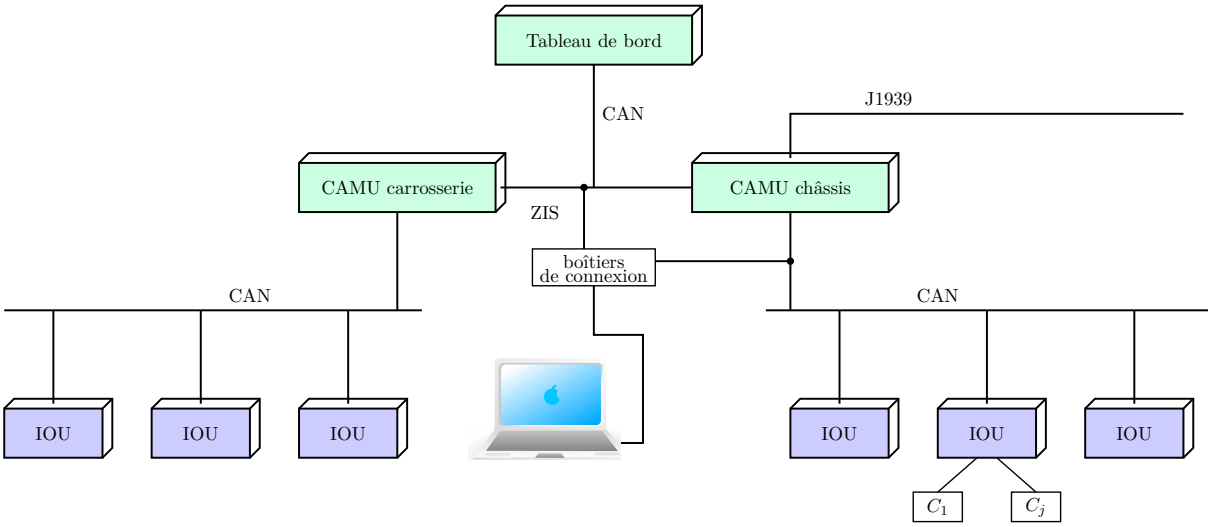


Figure 8.1 — Architecture Multibus

Les CAMUs (Central Management Unit) et le tableau de bord sont des calculateurs maîtres dans le sens où ce sont eux qui contiennent tous les programmes informatiques. Chacun gère un domaine différent :

- le tableau de bord gère tout ce qui est lié à l'interface homme machine avec le conducteur, notamment les commandes de l'utilisateur et l'affichage des différents voyants.
- le châssis : il gère la partie moteur, signalisation extérieure (feux de stop, de recul, indicateurs de directions, position, croisement...), essuie-vitres/lave-vitres, confort thermique, gestion de la rampe...
- la carrosserie : il gère tout ce qui est éclairage dans le véhicule, la gestion des portes, ou encore les demandes des passagers (demande d'arrêt, de sortie de rampe pour poussette ou pour handicapés)...

Les IOUs (Input Output Unit) sont des calculateurs esclaves et correspondent à des boîtes entrées/sorties reliées au réseau de communication. Ces calculateurs ont très peu d'intelligence embarquée. Ils récupèrent des valeurs logiques, analogiques, fréquentielles des capteurs qui leurs sont connectés et les émettent sur le réseau de communication. Ils pilotent les actionneurs selon les informations reçues par leur CAMU. Il peut y avoir jusqu'à dix calculateurs esclaves par CAMU. Le tableau de bord n'a pas d'IOU.

Chaque IOU est relié à un seul CAMU par un réseau de communication CAN. Le protocole CAN utilisé est le protocole de transmission de données le plus couramment employé dans l'industrie automobile. Les différents CAMUs peuvent également communiquer à travers un réseau CAN appelé Zone Inter-Systèmes (ZIS). Il existe donc trois réseaux CAN différents. Il existe un autre réseau CAN standardisé et noté J1939 qui permet le dialogue avec d'autres calculateurs développés par d'autres sociétés.

Pour surveiller le réseau de communication, un ordinateur portable y est connecté par l'intermédiaire d'un ou plusieurs boîtiers d'acquisition suivant le nombre de réseaux à surveiller. Un boîtier d'acquisition existant dans le commerce est utilisé de manière à obtenir les informations sous forme utilisable. Dans une version industrielle, l'objectif est d'embarquer le code dans les calculateurs, ou bien dans un calculateur dédié, ou encore dans un module embarqué qui pourra être ajouté ou retiré selon les besoins du garagiste.

Les informations, circulant sur le réseau CAN, sont contenues dans des trames de données. Ces trames sont envoyées de façon périodique (toutes les 50ms dans notre application) par les calculateurs. Chaque trame de données est une séquence de longueur fixe de bits. Elle commence par un identifiant unique, codé sur 16 bits. Cet identifiant permet, en autres, de connaître le calculateur qui émet la trame. Les trames envoyées par un calculateur sont toujours constituées par les mêmes variables. Seules leurs valeurs peuvent changer.

La valeur d'une variable est donc envoyée sur le réseau toutes les 50ms. Un modèle à événements discrets de la fonction est utilisé et par conséquent, seul un changement de valeur de ces variables est pris en compte. De surcroit, dans une trame de données, plusieurs variables peuvent être utiles et être reliées à des événements du modèle. Elles ne peuvent donc être ordonnées de manière séquentielle. Nous devons donc définir un lien entre ces variables et les transitions du réseau de Petri. C'est pourquoi, nous avons décidé d'associer ces variables par une conjonction logique dans le modèle.

Un événement tel que nous l'entendons dans les chapitres précédents, i.e. associé à une transition du modèle, peut être une conjonction logique de plusieurs variables. Nous ne considérons que les changements de valeur de ces variables que nous appelons événement atomique. Par exemple, le passage de 0 à 1 d'une variable v_1 est un événement atomique. Nous détectons l'insertion de tous les événements atomiques mais dans nos modèles, c'est le passage à vrai de la conjonction des variables qui permet le tir d'une transition et qui peut autoriser le changement de trajectoire. Lorsqu'une localisation est faite, c'est donc l'ensemble des variables associées à la transition qui est suspecté.

Pour résumer, il existe des variables v_i dont seul le changement de valeur est pris en compte. Le changement de valeur d'une variable est appelé événement atomique, noté e_i. Plusieurs événements atomiques peuvent être associés par une conjonction logique à une même transition t_i. C'est cette conjonction logique que nous appelons événement dans les chapitres précédents. Dès qu'il y localisation d'un événement i.e. d'une conjonction logique, les événements atomiques associés sont alors suspectés. Notons que nous pouvons également prendre en compte des conditions i.e. considérer la valeur et non plus le changement de celle-ci, dans les transitions. Mais aucune détection n'est faite sur ces conditions. Par exemple, les portes d'un autobus ne peuvent être ouverte si la vitesse du véhicule n'est pas nulle.

Nous avons donc pu avoir accès à un banc de tests représentant l'architecture d'un autobus de ville (figure 8.2 et 8.3).

8.1.2 Le diagnostic dans les véhicules de transport

Sur le système étudié, il existe des fonctions de diagnostic qui permettent de détecter et localiser des défauts. Il s'agit de détecter l'apparition et la disparition :

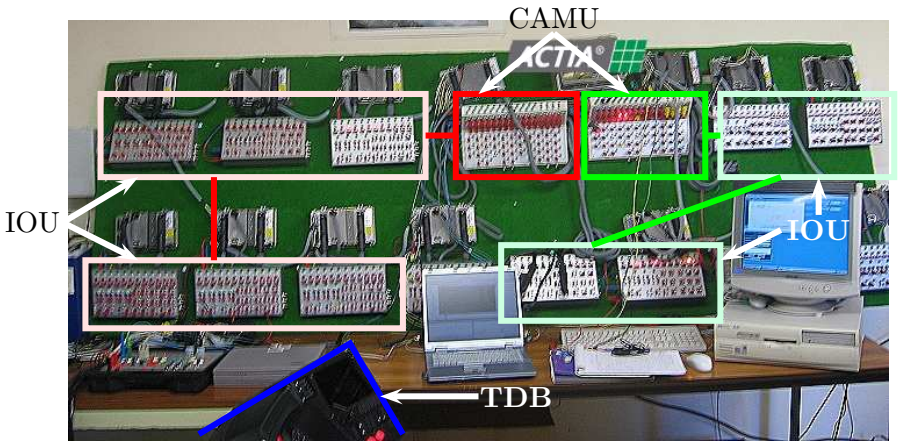


Figure 8.2 — Banc de test 1

Figure 8.3 — Banc de test 2

- des défauts internes aux calculateurs CAMU et aux boîtiers IOU,
- des défauts de réseau de communication entre les différents boîtiers,
- des défauts sur les sorties du système et sur les entrées analogiques et fréquences non utilisées en entrées logiques.

Défauts internes au calculateur CAMU et aux boîtiers IOU Les boîtiers IOU sont capables de détecter une surchauffe de leur boîtier. Ils peuvent également détecter une défaillance d'une partie de l'électronique interne du boîtier.

Les boîtiers CAMU surveillent en permanence leurs réseaux de communication avec les autres boîtiers et remontent les défauts suivants :

- défaut de communication avec le Tableau de bord,
- défaut de communication avec les boîtiers IOU,
- défaut de communication avec l'autre boîtier CAMU,

Défauts de communication Ils sont immédiatement détectés lorsqu'une défaillance du réseau de communication (coupure d'alimentation, défaut interne) apparaît ou alors lorsqu'il y a un défaut de câblage ou une perturbation.

Les boîtiers IOU signalent également au boîtier CAMU tout défaut sur leur alimentation.

Défauts sur les sorties du système Chaque sortie des boîtiers CAMU et IOU possède trois types de détection :

- circuit ouvert (CO) qui peut être dû à un câblage défaillant ou à un actionneur hors service (ampoule grillée par exemple),
- court-circuit (CC) qui peut être dû à un câblage défaillant ou à un actionneur hors service en court-circuit,
- Échauffement trop important du composant de sortie qui peut être dû à une surcharge trop importante ou à une autoprotection du composant.

Certains défauts se manifestant par ces CO, CC ne sont pas détectés. Par exemple, la conception des entrées logiques (ici des interrupteurs) et le principe de câblage ne permet pas de détecter les défauts sur celles-ci.
La détection et la localisation de ces défauts posent donc des problèmes. C'est donc sur ce type de défauts que nous allons nous concentrer.

8.1.3 Une fonction : la fonction d'accès rampe pour handicapés

Actia développe les différentes fonctions retrouvées dans un autobus comme par exemple la gestion des portes ou encore la gestion de la rampe d'accès pour handicapés. Nous avons pu obtenir les différentes données de conception relatives à diverses fonctions du véhicule. Notre choix s'est porté sur la fonction "gestion de la rampe d'accès pour handicapés" car cette fonction a une complexité moyenne (par rapport à celle de "la gestion des portes") qui suffit pour valider notre approche. Cette fonction a besoin des informations provenant des trois CAMUs. Néanmoins, l'observation de deux CANs suffit pour avoir l'ensemble des variables utiles pour cette fonction.

Finalement, le schéma électrique de la fonction est décrit sur la figure 8.4.

Pour simuler la rampe, un moteur est utilisé avec une touillette qui tourne à la manière d'une montre. Deux contacteurs simulent les capteurs de rentrée et de sortie de la rampe.

Nous avons modélisé le comportement de cette fonction par un réseau de Petri. Pour cela, nous avons utilisé dans un premier temps le cahier des charges de cette fonction puis nous avons modifié ce modèle en nous appuyant sur le code développé afin d'avoir un modèle le plus proche possible de celui implémenté.

La fonction rampe est gérée principalement par le châssis mais a besoin des informations provenant de la partie carrosserie et du tableau de bord.

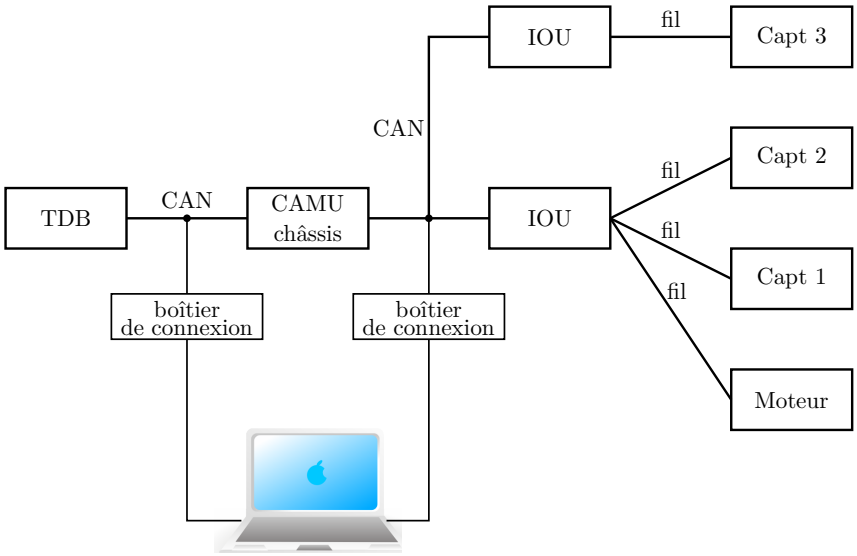


Figure 8.4 — Schéma électrique pour la fonction rampe

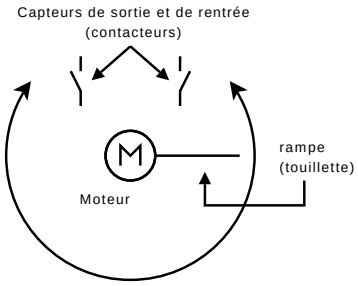


Figure 8.5 — Construction de la rampe

Fonctionnement de la rampe

Pour pouvoir sortir la rampe, il est nécessaire que la porte où la rampe est installée soit fermée et que le véhicule soit à l'arrêt.

La sortie de la rampe est effectuée lorsqu'une demande de sortie est émise et que les conditions citées ci-dessus sont remplies. La sortie de la rampe ne pourra s'effectuer que si le conducteur appuie de façon continue sur la demande de sortie. Si celle-ci disparaît alors que la rampe n'est pas totalement sortie alors la rampe devra rentrer. Elle peut ressortir dès la commande suivante de sortie de rampe.

Lorsque la rampe est totalement sortie, ce qui est indiqué par le capteur de sortie de la rampe, alors le moteur n'est plus alimenté.

Pour rentrer la rampe, les conditions sont les mêmes que pour la sortie de la rampe mise à part que celle-ci ne doit pas être rentrée c'est-à-dire que le capteur de rentrée de rampe n'est pas vrai.

La rentrée de la rampe est effectuée lorsqu'une demande de rentrée est émise et que les conditions citées ci-dessus sont remplies.

Le moteur de la rampe est alors piloté en marche arrière.

Même si le signal de demande de rentrée de rampe disparaît, la rampe doit continuer à rentrer jusqu'à ce que le capteur de rentrée passe à vrai (sauf si une demande de sortie est émise).

Lorsque la rampe est rentrée, alors le moteur n'est plus alimenté.

Représentation de la rampe

Une représentation de la rampe est donnée sur la figure 8.6. Ce modèle n'est constitué que d'événements observables puisque le cahier des charges utilisé décrit uniquement les communications réseaux entre les différents CAMUs et entre les CAMUs et leurs IOUs.

Six variables sont visualisées sur le réseau de communication :

Événement / Signification	Mnémonique
v_1 : commande du moteur en marche arrière	CHA_SMRAMPR
v_2 : commande du moteur en marche avant	CHA_SMRAMPS
v_3 : demande de la sortie de la rampe	TDB_ESRAMP
v_4 : demande de la rentrée de la rampe	TDB_ERRAMP
v_5 : capteur de sortie de la rampe	ESROR
v_6 : capteur de rentrée de la rampe	ERREN

À partir de ces variables, nous définissons neuf événements :

Événement / Signification	Mnémonique	Valeur attendue
e_1 : mise en marche arrière du moteur	CHA_SMRAMPR	0x01
e_2 : arrêt du moteur en marche arrière	CHA_SMRAMPR	0x00
e_3 : arrêt du moteur en marche avant	CHA_SMRAMPS	0x00
e_4 : mise en marche avant du moteur	CHA_SMRAMPS	0x02
e_5 : arrêt de la demande de sortie de la rampe	TDB_ESRAMP	0x00
e_6 : demande de la sortie de la rampe	TDB_ESRAMP	0x10
e_7 : demande de la rentrée de la rampe	TDB_ERRAMP	0x08
e_8 : capteur de rampe sortie	0ESROR	0x00
e_9 : capteur de rampe rentrée	0ERREN	0x00

Nous représentons sur ce modèle les différentes phases du comportement de la rampe que nous avons citées ci-dessus et qui peuvent se résumer comme :

- la rentrée de la rampe si demande,
- la sortie de la rampe si demande,
- la rentrée de la rampe si demande de sortie disparait.

Remarque : la place P_3 correspond au fait que le conducteur appuie sur la demande de sortie alors que la rampe est déjà sortie (tir de la transition t_{12}) et prend en compte le relâchement du bouton de la demande de sortie (tir de la transition t_{13}). Le relâchement du bouton de demande de sortie doit être considéré car cet événement a des répercutions sur le fonctionnement de la rampe (lors de la sortie de la rampe). Il doit donc être pris en compte dans ce cas là.

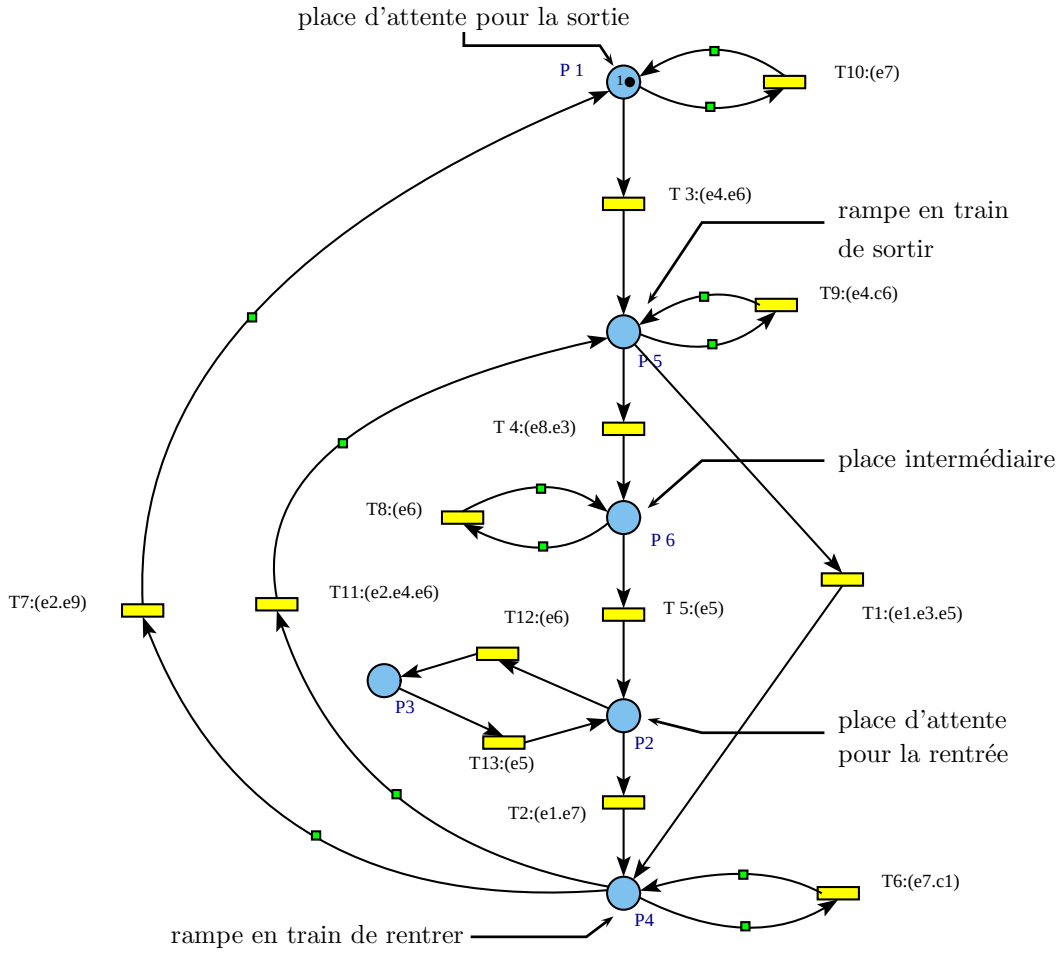


Figure 8.6 — Représentation de la fonction rampe

Nous avons rajouté des demandes sur l'ensemble des états car nous avons pris en compte le fait que le conducteur puisse actionner les demandes à n'importe quel moment. Deux conditions c_1 et c_6 ont été ajoutées. c_1 correspond au fait que le moteur est piloté en marche arrière ($CHA\_SMRAMPR = 0x01$) et c_6 correspond à $TDB\_ESRAMP = 0x01$. Par exemple, la transition T_6 correspond au fait que le conducteur peut à nouveau faire une demande de rentrée de rampe alors que la rampe est déjà pilotée en marche arrière.

De plus, certaines transitions comme $T_1 : (e_1.e_3.e_5)$ ont plusieurs variables associées car les variables se trouvent dans la même trame (comme c'est le cas de e_1 et e_3). Nous avons également décidé d'associer des variables à une même transition même si elles ne font pas partie de la même trame. Il s'agit de variables dont les événements atomiques sont liés. Notamment, lorsqu'une demande est envoyée, celle-ci est généralement suivie par une commande. Deux messages sont envoyés sur le réseau. Si nous associons chaque variable à une transition différente et si une absence (respectivement une insertion) de la demande implique une absence (respectivement une insertion) de la commande alors nous ne respectons plus nos hypothèses à savoir l'insertion ou l'absence d'un seul événement. En associant ces variables par une conjonction logique à une transition, nous respectons à nouveau nos hypothèses. Par contre, cela rend notre localisation moins précise puisque nous suspecterons plus de variables.

8.2 Applications développées

8.2.1 Structure générale

Pour notre application, un outil intéractif permettant d'éditer les réseaux de Petri est nécessaire. Nous avons donc décidé de réutiliser un logiciel libre existant appelé Roméo que nous allons modifier pour les besoins de l'application.

Le logiciel Roméo est développé par l'équipe Système Temps Réel de l'Institut de Recherche en Communications et en Cybernétique de Nantes (IRCCyN) (http ://romeo.rts-software.org) que nous remercions pour avoir à disposition les sources de leur projet. Roméo comporte une interface graphique (écrit en TCL/Tk) afin d'éditer et de concevoir des réseaux de Petri, et des modules dédiés aux calculs (Mercutio, écrit en C++). Il permet notamment une simulation des réseaux de Petri. C'est lui qui va gérer toute la partie réseau de Petri et la partie IHM (Interface Homme Machine).

Nous avons également besoin d'une application qui va gérer l'interfaçage entre le matériel et l'application Roméo. **L'application DDP** a donc été développée et permet :
- de récupérer les trames dans les boîtiers d'acquisition,
- d'extraire de ces trames les données utiles,
- de vérifier si ces données permettent le franchissement d'une transition sensibilisée,
- et de communiquer avec Roméo.

La communication avec Roméo consiste à lui fournir les transitions qui doivent être franchies. Quant à Roméo, il fournit à DDP les transitions qui sont sensibilisées (et par conséquent les événements qui leurs sont associés).

Nous pouvons visualiser sur la figure 8.7 la structure globale de ces applications.

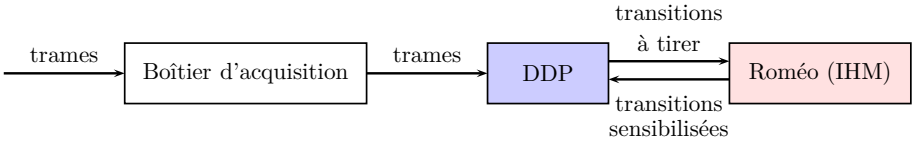


Figure 8.7 — Structure générale des applications

Nous allons développer un peu plus ces deux applications. Une représentation plus détaillée de la structure des applications est donnée sur la figure 8.9.

8.2.2 L'application Roméo

Une première partie de l'application Roméo gère toute la partie IHM proprement dite. Cette partie IHM peut être visualisée sur la figure 8.8. Les modèles sont construits grâce à cette interface qui permet d'ajouter de nouvelles places, transitions et arcs. Dans la fenêtre de simulation sont représentées les différentes actions de franchissement et les nouvelles tran-

sitions sensibilisées. Dans sa forme initiale, la simulation se faisait à partir des clics de souris sur les transitions sensibilisées. Nous avons modifié l'application pour qu'elle communique avec notre application DDP. Roméo reçoit les transitions à tirer de la part de DDP. Dès réception, ces transitions sont tirées de façon automatique. Il est toujours possible de cliquer sur les transitions pour forcer le tir.

Nous avons également créé deux nouvelles fenêtres, l'une qui indique les différents messages qui sont échangés entre Roméo et DDP (partie Communication) et l'autre qui indique toutes les actions qui sont effectuées dans Roméo (partie Console IHM).

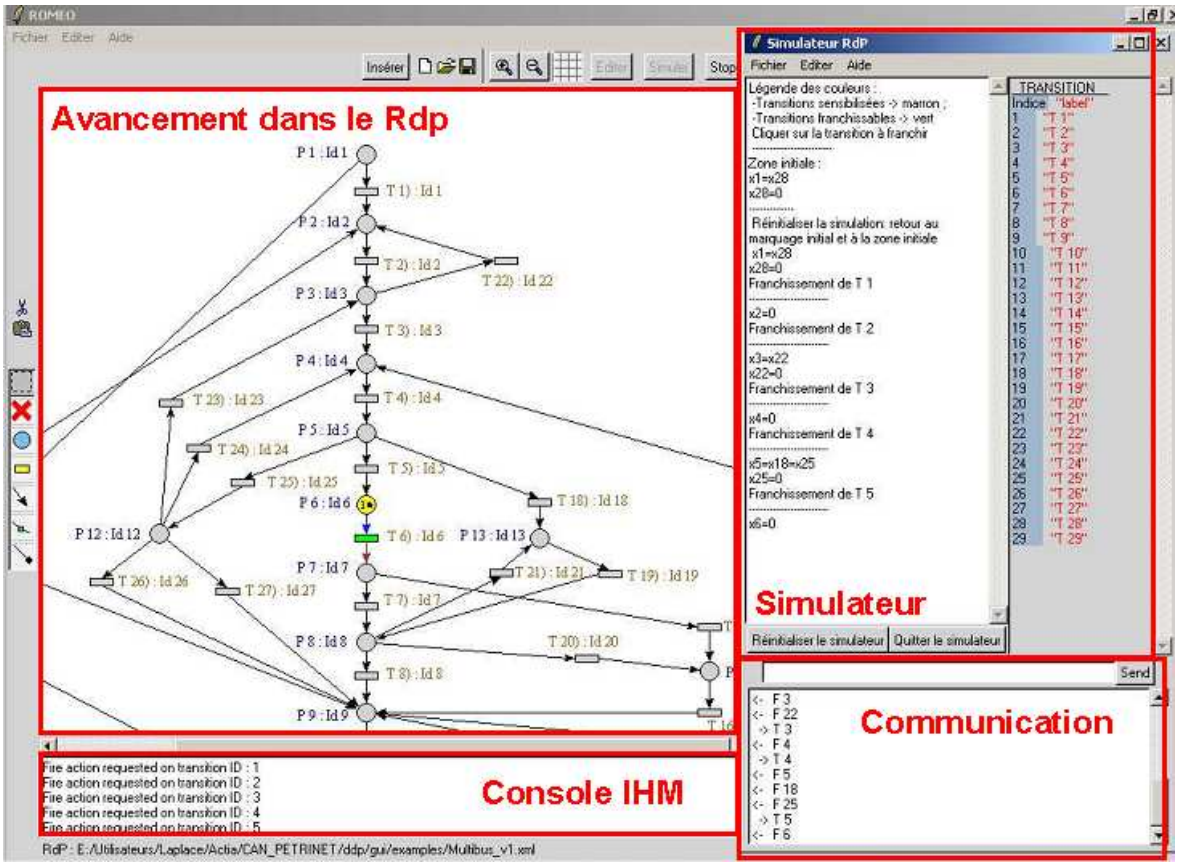


Figure 8.8 — Interface graphique de Roméo

C'est dans cette application qu'est définie entièrement la fonction. Son comportement est représenté par les réseaux de Petri. Mais nous avons vu que les événements sont associés aux transitions permettant ainsi de faire évoluer le modèle. Ces événements sont constitués à partir des variables contenues dans les trames de données.

Nous avons donc modifié le logiciel Roméo pour que nous puissions ajouter à partir d'une fenêtre les différentes trames ainsi que les différentes variables (notées dans l'application "événements") qui nous intéressent.
À ces variables, nous leur associons :

- une trame : identifiant et initialisation des données,
- un nom,
- un ByteMaskNB, i.e. l'octet où se trouve la variable que nous souhaitons récupérer

dans la trame (dans notre application, nos trames ont huit octets),
- un mask, i.e. un masque en hexadécimal que nous appliquons à l'octet pour regarder le ou les bits qui nous intéressent (et donc la variable),
- et enfin un code en hexadécimal, i.e. une valeur qui correspond à un front montant (code valant un) ou un front descendant (code valant zéro).

Exemple : Nous associons une variable v_1 à une transition. Un front montant sur cette variable permet le tir de la transition. Cette variable se trouve dans la trame N (définie en hexadécimal) :

N : 00 01 0f fa ff 12 50

Cette variable se situe au niveau du second bit du deuxième octet de la trame. Le ByteMaskNB vaut donc deux comme le mask. C'est le front montant qui est attendu. Le code vaut donc un. Si nous appliquons ces différents masques à la trame précédente, nous obtenons une valeur de v_1 égale à 1. C'est cette valeur que nous comparons au code que nous avons défini précédemment. Pour tirer la transition, il faut que la valeur obtenue soit égale à notre code mais également que la précédente valeur de la variable soit à zéro (front montant). L'application Roméo dédie le calcul de l'évolution du réseau de Petri à la DLL Mercutio. C'est cette application qui va, à partir des transitions à tirer et du marquage courant, calculer le marquage suivant et les nouvelles transitions sensibilisées qui sont de nouveau envoyées à Roméo pour qu'il mette à jour le modèle.

Pour résumer, nous pouvons dans le logiciel Roméo décrire le comportement d'une fonction sous forme de réseau de Petri. Ce comportement évolue en fonction des messages qui circulent sur le réseau de communication. L'ensemble de ces informations est également adjoint au modèle. Dans une phase d'initialisation, Roméo va envoyer à l'application DDP toutes les informations concernant sa structure statique, i.e. le nombre de places, de transitions, l'association événements-transitions, toutes les informations sur les variables... Dans la phase de simulation, Roméo envoie à DDP l'ensemble des transitions sensibilisées. Notons qu'il est nécessaire d'envoyer seulement les noms des transitions puisque DDP connaît déjà les événements associés aux transitions.

8.2.3 L'application DDP

Les programmes développés peuvent être décomposés en deux sous parties : une partie qui va acquérir les trames, les filtrer et extraire les données, puis une autre partie qui va gérer le réseau de Petri et l'affichage, sur une console, du suivi des trames, des événements reçus, de la détection... L'application DDP est composée de programmes faisant partie de ces deux sous ensembles (voir figure 8.9).

L'application DDP est donc composée de deux processus : l'un gère la lecture et le traitement des trames du CAN et l'autre gère la demande de tir des transitions.

Processus CanListener Dans un premier temps, l'application doit récupérer les trames de données contenues dans le boîtier d'acquisition. Ce boîtier prévient l'application lorsqu'il reçoit des nouvelles trames qui sont ensuite chargées par DDP dans un buffer, c'est-à-dire un

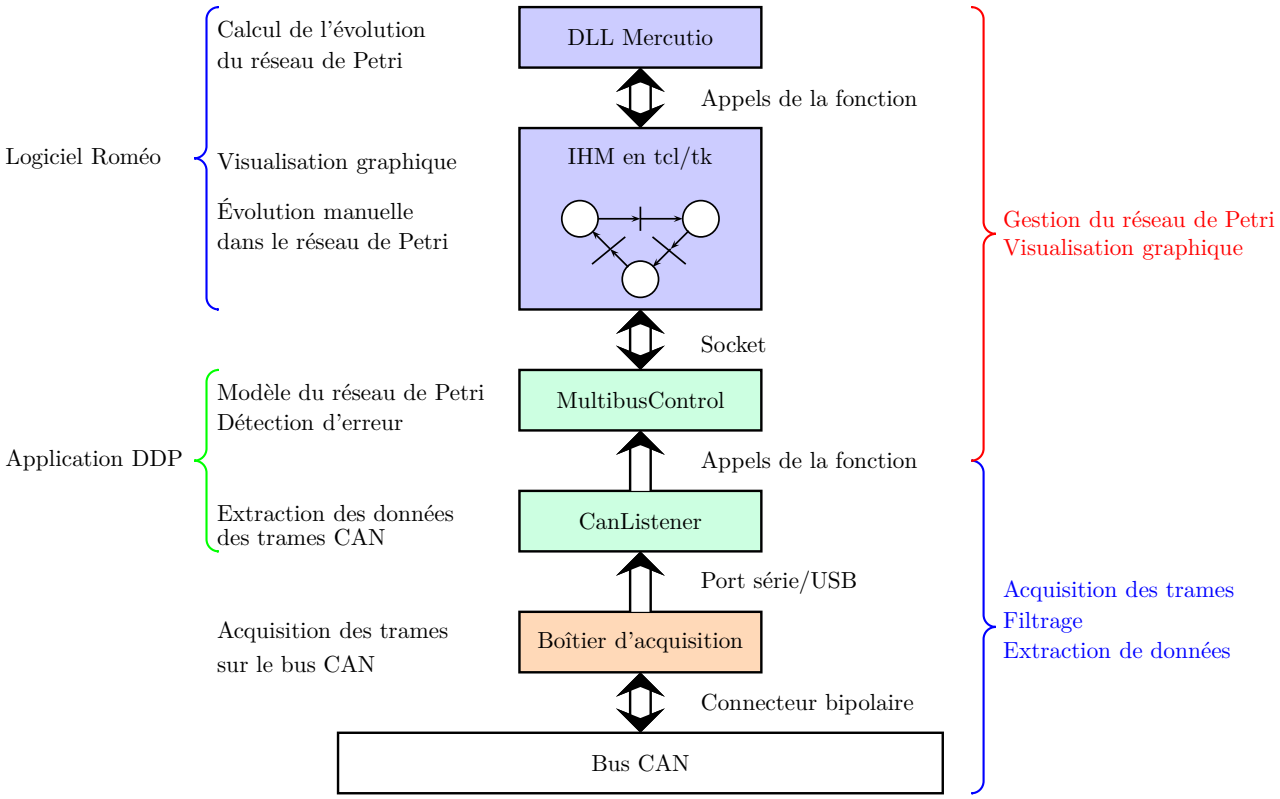


Figure 8.9 — Conception du programme

espace de stockage. Ensuite ces trames sont à nouveau filtrées pour considérer seulement les trames que nous voulons surveiller. En effet, le premier filtre est un masque et par conséquent il laisse passer toutes les trames qui correspondent à ce masque même si ces trames n'apportent aucune information utile pour notre modèle.

L'ensemble des trames qui circulent sur le réseau peut être visualisé sur la console du programme (leur identifiant et leurs données). Cette visualisation peut également être limitée aux trames qui nous intéressent (figure 8.10).

Figure 8.10 — Visualisation des trames

À partir de ces trames, les données contenues dans celles-ci sont récupérées et sont transmises au programme MultibusControl par une "mise à jour" de la trame.

Processus MultibusControl Le processus MultibusControl est tout d'abord initialisé par le modèle provenant de Roméo. L'application DDP charge en fait le fichier de Roméo écrit en XML, pour récupérer toutes les informations qui lui sont utiles. Ainsi, les associations trames-variables-transitions sont définies dans DDP et seront par la suite utilisées par le programme MultibusControl.

Lorsqu'une trame est identifiée, le programme effectue dans l'ordre les tests suivants :

1. il recherche les variables qui lui sont associées,
2. pour chaque variable appartenant à cette trame, il regarde s'il y a eu un changement de valeur de cette variable, i.e. un événement atomique,
3. si c'est le cas, alors il regarde les transitions qui lui sont associées,
4. il regarde si elles sont sensibilisées,
5. s'il n'existe qu'une seule variable associée à cette transition alors il envoie une demande de tir à Roméo,
6. s'il existe plusieurs événements associées, le programme doit attendre un temps prédéfini (200ms dans notre cas) les autres événements. Pour cela, un processus est lancé et s'endort pendant le temps prédéfini. Si tous les événements sont reçus, le processus précédent est détruit, sinon lors de son réveil, il lance un message de détection.

Le programme MultibusControl détermine les transitions qu'il faut tirer afin de les envoyer à Roméo pour qu'il puisse simuler l'avancement des jetons dans le réseau de Petri. C'est la procédure de surveillance. La détection se fait dans le même temps. En effet, il y a deux endroits où une détection peut se faire lors de cette phase de surveillance :

1. lorsque l'événement n'est associé à aucune transition sensibilisée,
2. lorsque le processus d'attente est fini alors que toutes les variables n'ont pas été reçues.

C'est donc l'application DDP qui va se charger de détecter si un événement ne correspond pas à celui qui est attendu. Le mécanisme général est présenté sur la figure 8.12.

Différentes informations comme la réception d'un événement, ou encore l'envoi d'une transition... peuvent être également visualisées sur la console du programme. C'est sur cette console que la détection d'un événement incohérent est visualisée (figure 8.11).

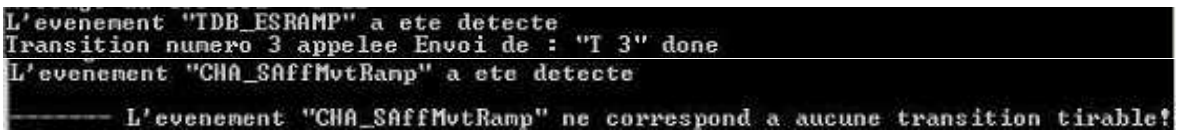


Figure 8.11 — Visualisation sur la console d'un tir de transition et d'une détection

Nous avons développé toutes les applications nécessaires pour suivre en ligne l'évolution d'un réseau de Petri. Il faut maintenant construire ce réseau de Petri. Le modèle de bon comportement est construit par un expert mais la construction des différents modèles vus dans le chapitre 7, à savoir $R^+, R^-, R^+_{BF}, R^+_{SYNC}, R^+_{DIAG}, \ldots$, est faite automatiquement comme nous allons le voir maintenant.

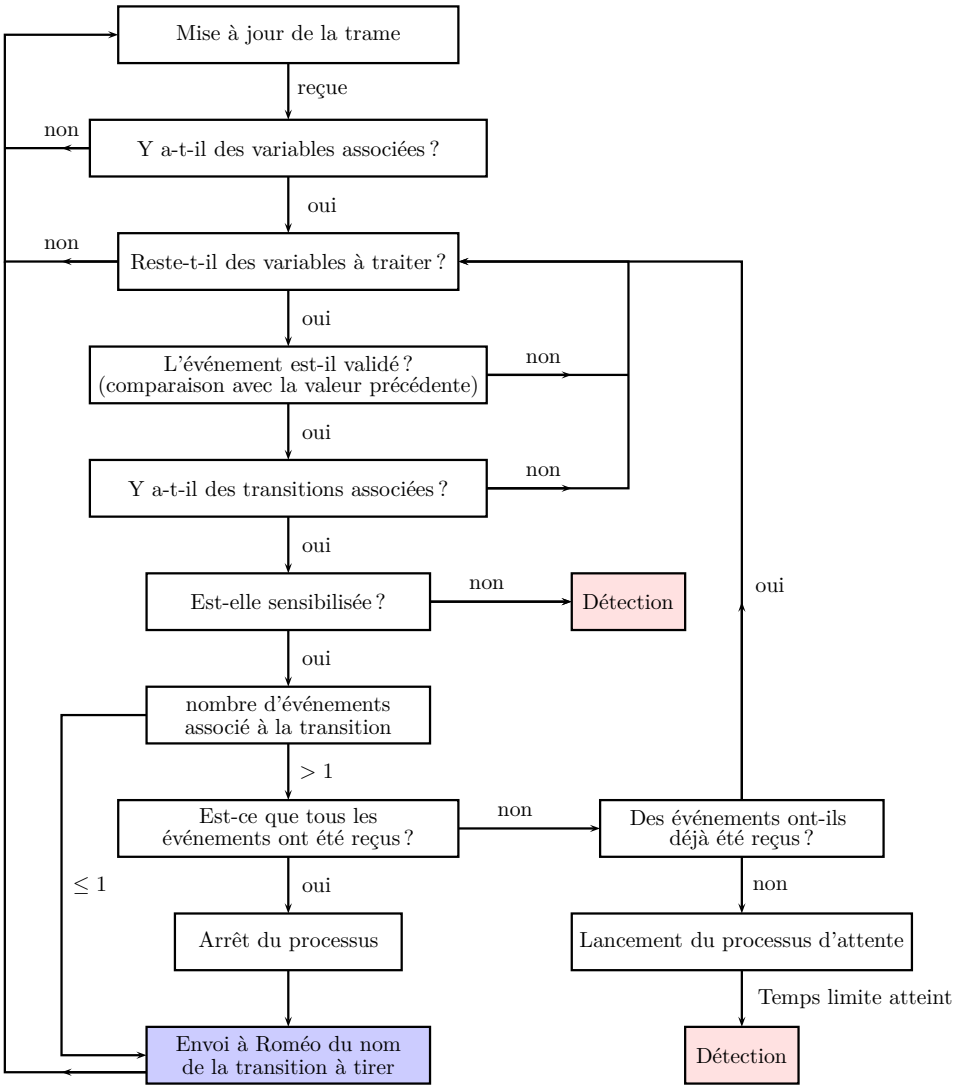


Figure 8.12 — Mécanisme général de l'application DDP

8.2.4 Construction des différents modèles

Les applications précédentes permettent de visualiser l'évolution du comportement d'une fonction sur Roméo, ainsi que de détecter une incohérence entre les événements reçus et ceux qui sont attendus par le modèle.

En ce qui concerne la phase de localisation, il nous faut construire les diagnostiqueurs, sous forme de réseaux de Petri que nous pourrons voir évoluer sur le logiciel Roméo. Ainsi, lors de la détection d'une incohérence, il suffira de regarder l'état dans lequel se trouve le réseau de Petri pour connaître les hypothèses de fautes qui sont susceptibles d'expliquer les observations reçues.

Pour construire ce diagnostiqueur, nous avons développé un programme qui, à partir de réseau de Petri de bon comportement, calcule de façon automatique les diagnostiqueurs sous format XML que Roméo pourra réutiliser. Pour cela, les différents modèles sont construits

étapes par étapes (voir figure 8.13) :

1. calculs des modèles de classes de fautes à partir du modèle de bon comportement,
2. calculs des modèles incluant les défaillances génériques à partir des modèles de bon comportement et des modèles de classes de fautes,
3. calculs des modèles de synchronisation à partir des modèles de classes de fautes et des modèles incluant les défaillances génériques,
4. calculs des diagnostiqueurs à partir des modèles de synchronisation.

Les modèles de classe de fautes peuvent être modifiés/construits par un expert. Dans notre cas, nous les construisons de façon automatique car nous suspectons l'insertion ou l'absence de tous les événements.

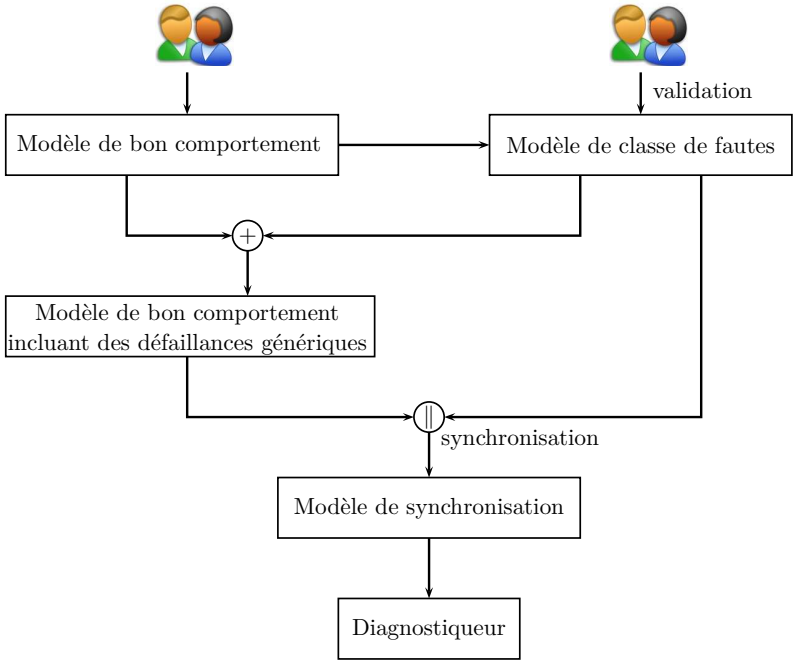


Figure 8.13 — Construction des différents modèles par l'algorithme

8.3 Scenarii de fautes

8.3.1 Limite du diagnostic des véhicules de transport

Dans un premier temps, si nous considérons le diagnostic qui est effectué dans les autobus, celui-ci ne prend pas en compte les défauts sur les capteurs logiques tels que les capteurs de rampe rentrée ou rampe sortie. Ainsi, aucune détection ne sera faite si un défaut intermittent se produit sur l'un ou l'autre des capteurs et ce, dans n'importe quel état de la fonction.

8.3.2 Détection et localisation de base

Avec notre surveillance, nous pouvons détecter la défaillance intermittente dans la plupart des états du modèle. En effet, l'insertion de l'événement "rampe rentrée" par exemple (c'est-

à-dire le passage de 0 à 1 du capteur de rentrée) est immédiatement détectée s'il n'est pas attendu. Grâce à cette surveillance, nous apportons un plus au diagnostic actuel.

8.3.2.1 Défaut sur le capteur de rentrée de la rampe

Si, maintenant, l'insertion de l'événement "rampe rentrée" (e_9) a lieu dans la place P_4 qui correspond à la place de rentrée de rampe alors la commande de rentrée de rampe s'arrêtera (e_2) puisqu'il s'agit d'un événement attendu. La rentrée de la rampe est stoppée et la rampe reste au milieu alors que, pour le modèle, la rampe est rentrée.

Le défaut étant intermittent, nous supposons ici qu'il a disparu lorsque le conducteur réagit. Le conducteur appuie alors de nouveau sur la demande de rentrée de la rampe (e_7) et une commande de rentrée rampe (e_1) est de nouveau envoyée.

Étant dans la place qui correspond à une rampe rentrée, l'insertion de l'événement $c_1.e_7$ est par conséquent détectée alors que c'est l'événement $e_2.e_9$ associé à la transition T_7 qui s'est réellement inséré. Le réseau de Petri de la fonction ne suffit donc pas pour localiser correctement.

8.3.3 Localisation à partir du diagnostiqueur

Nous représentons sur la figure 8.14 le diagnostiqueur associé à l'insertion.

8.3.3.1 Défaut sur le capteur de rentrée de la rampe

Si nous reprenons le diagnostiqueur, le modèle évolue normalement vers l'état $((P_1, Ok)\ldots)$ à partir de l'état $((P_4, Ok)\ldots)$ et de la réception de l'événement inséré $e_2.e_9$. lorsque celui-ci se trouve dans l'état $((P_1, Ok)\ldots)$, la réception de l'événement $c_1.e_7$ le fait passer dans l'état $(P_4, F_{t_7^+})$. Cet état est associé à l'hypothèse que l'événement $e_2.e_9$ lié à T_7 s'est inséré. L'insertion des événements atomiques e_2 et e_9 est donc suspectée. Il y a l'événement recherché à savoir l'événement "capteur de rampe rentrée" i.e. e_9. L'événement incohérent à savoir l'événement atomique e_7 est également suspecté.

8.3.3.2 Défaut sur le capteur de sortie de la rampe

Pour la sortie de la rampe, le conducteur doit appuyer de façon continue sur la demande de sortie de la rampe (e_6). Si un défaut apparaît sur le capteur de sortie de rampe alors que le modèle se trouve dans l'état $((P_5, Ok)\ldots)$, alors l'événement associé attendu (e_8) fait évoluer le modèle dans l'état $((P_6, Ok)\ldots)$ avec l'arrêt de la commande (e_3). Lorsque le défaut disparaît, une nouvelle commande est envoyée (e_4) et le modèle évolue dans l'état $(P_5, F_{t_4^+})$. À cet état est associée l'hypothèse que l'événement $e_3.e_8$ lié à T_4 s'est inséré. Les insertions des événements atomiques e_3 et e_8 sont suspectées. Il y a l'événement recherché à savoir l'événement "capteur de rampe sortie" i.e. e_8. L'événement incohérent à savoir e_4 est également suspecté.

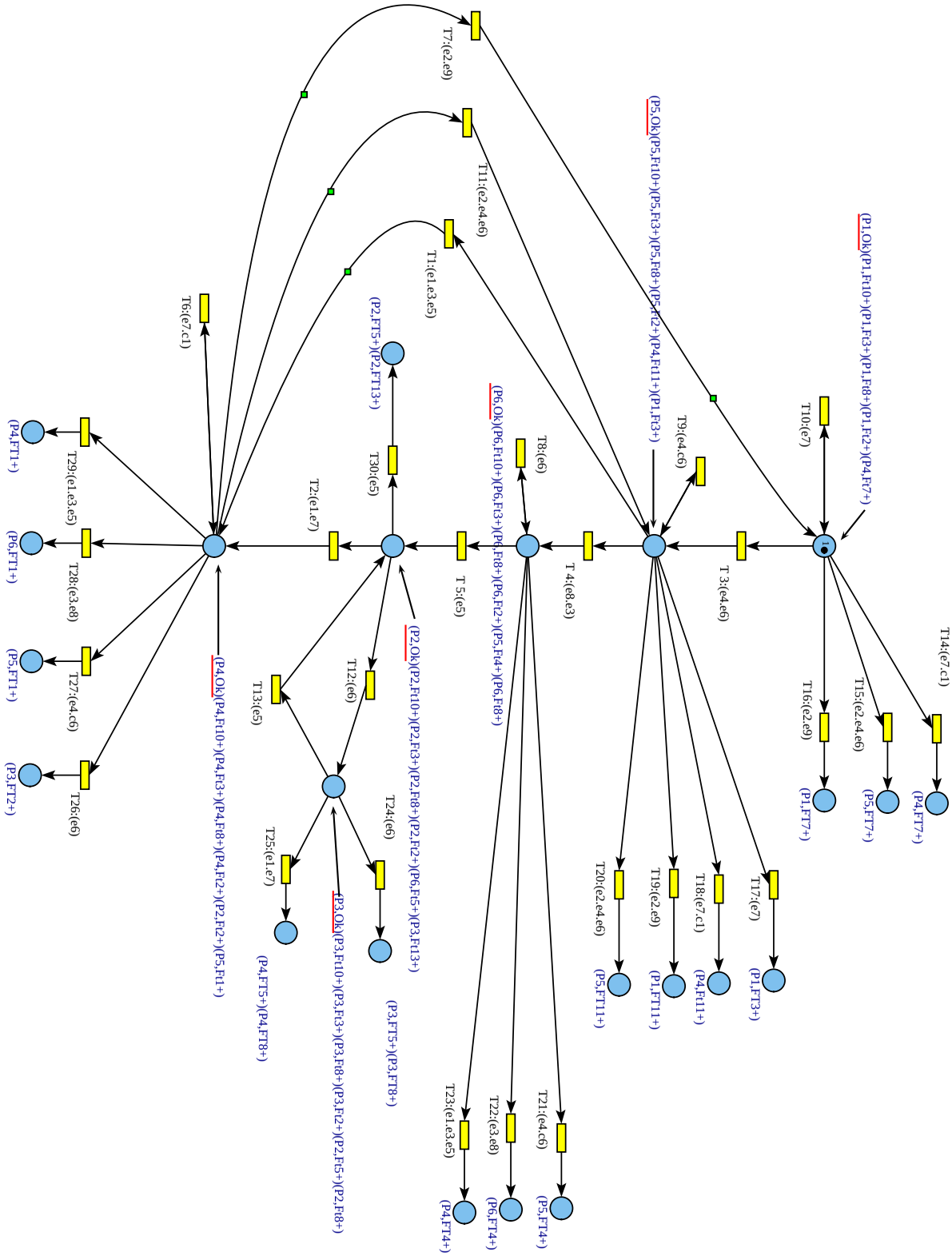


Figure 8.14 — Diagnostiqueur insertion pour la fonction rampe

8.3.3.3 Limitations

Ceci ne fonctionne que si le conducteur continue d'appuyer sur la commande lorsque le défaut disparaît. Si le conducteur arrête d'appuyer sur la demande (e_5) alors que le défaut n'a pas disparu, le nouvel état devient $((P_2, Ok)\ldots)$. En effet, nous avons dû prendre en compte le relâchement de la demande puisqu'elle agit directement sur le système. Quand la rampe sort, si la demande disparaît alors la rampe doit rentrer. Il faut donc prendre en compte explicitement cet événement à savoir le front descendant de la demande de sortie de rampe. Nous ne pouvons pas associer cette variable à la transition précédente car le conducteur relâche la demande quand il veut. Il y a donc insertion de deux événements : celui associé à la transition t_4 (insertion de l'événement associé au capteur de sortie) et celui associé à la transition t_5 (relâchement du bouton). On sort donc de nos hypothèses.

Néanmoins, connaissant cette particularité, les modèles de classe de fautes peuvent être modifiés par un expert pour que le diagnostiqueur puisse prendre en compte ce défaut. Il s'agit d'intégrer un modèle de faute à partir de la connaissance du système. Pour cela, une transition t_5^+ est rajoutée au modèle à la suite de t_4^+.

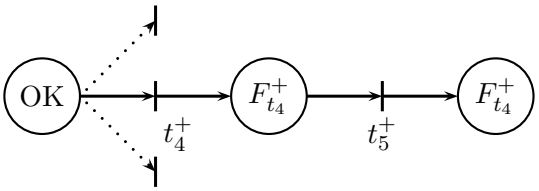


Figure 8.15 — Modification apportée au modèle de classe de faute

Dans le modèle de bon comportement modifié, cela se traduit par la création d'une transition t_5^+ bouclée sur P_5 (rampe en train de sortir). Ces deux transitions t_5^+ et t_4^+ représentent l'insertion d'événement alors que le modèle était dans la place P5. Nous voulons donc rester dans cette place malgré la réception de ces deux événements.

Dans le diagnostiqueur, cela se traduit par une duplication des états $(P_5, F_{t_4^+})$, $(P_6, F_{t_4^+})$, $(P_4, F_{t_4^+})$ et des transitions T_{21}, T_{22}, T_{23} qui auraient pour place d'entrée la place $((P_2, Ok)\ldots)$ (figure 8.16).

Malgré cela, cette méthode est coûteuse, longue et nécessite l'intervention d'un expert. Nous retrouvons ici une limitation due à nos hypothèses initiales qui prennent en compte l'insertion ou l'absence d'un seul événement. Il faudrait par la suite développer cette approche en considérant l'insertion ou l'absence d'une séquence d'événements.

8.4 Conclusion

Ce chapitre présente une plateforme logicielle (DDP et Roméo) dédiée à la surveillance du comportement d'une fonction à travers un réseau de communication et ce, dans un contexte embarqué. Cette plateforme permet aussi de détecter et de localiser les défaillances intermittentes se traduisant par l'insertion ou l'absence d'un événement sur le réseau de communication.

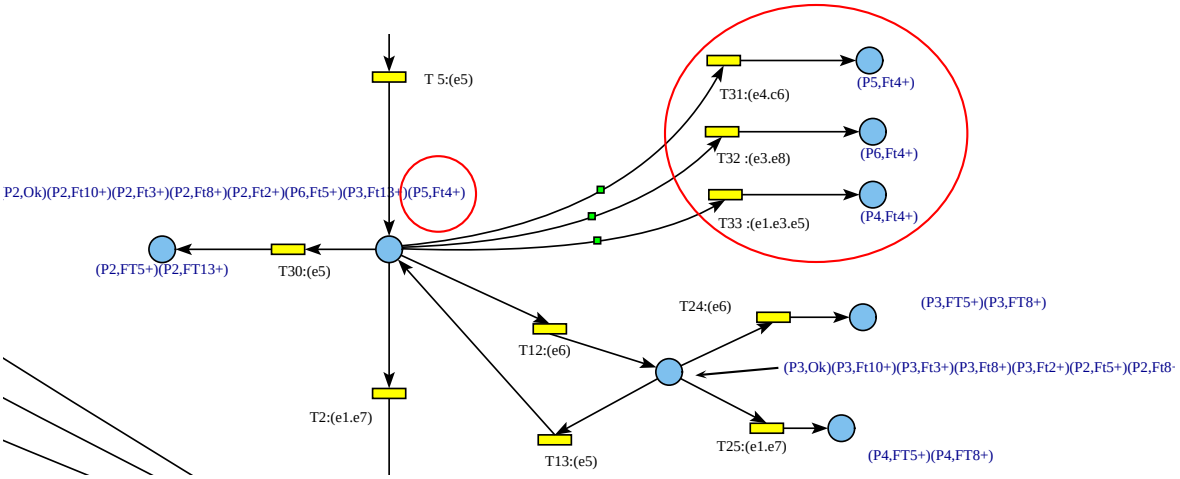


Figure 8.16 — Diagnostiqueur intégrant un modèle de faute spécifique

Nous avons également développé un programme qui permet à partir du modèle de bon comportement d'une fonction d'obtenir les diagnostiqueurs spécifiques à l'insertion ou l'absence d'un événement qui permettrons de localiser ces défaillances.

Ces différents outils ont été utilisés sur une application du domaine industriel : un banc de test représentant l'architecture électronique utilisée dans un autobus de ville. Les défaillances retrouvées dans ces transports peuvent porter sur les capteurs dont les manifestations par des circuits ouverts ou des courts-circuits correspondent à un mode de fonctionnement du système, ce qui les rend difficiles à diagnostiquer. Ceci est dû en partie à une vision limitée et non globale du système. Notre approche permet de prendre en compte le fonctionnement global du système, et par conséquent, de détecter et localiser ce genre de défaillances.

Notre approche, validée sur une application réelle, apporte un réel avantage au diagnostic effectué actuellement dans les systèmes de transport. Des défaillances intermittentes, difficiles à déterminer, sont détectées et localisées. Cela nous a permis également d'en déterminer les limites qui pourront faire l'objet d'une étude plus approfondie dans une perspective de ces travaux.

Conclusions, discussions et perspectives

8.1 Conclusions

Les travaux présentés dans cette thèse traitent du diagnostic de défaillances intermittentes dans les systèmes à événements discrets.

Nous nous focalisons sur les défaillances qui se traduisent par l'insertion ou l'absence d'un événement dans le système. Ces événements fautifs ne sont pas forcément immédiatement détectés car ils peuvent correspondre en temps normal à des événements de bon comportement.

L'approche proposée est une approche basée cohérence s'appuyant sur des modèles de bon comportement du système. Le principe de détection consiste à comparer les événements attendus par le modèle et ceux observés. Dès qu'une incohérence existe entre ces événements, cela traduit la présence d'une défaillance. C'est l'étape de détection.

Nous avons ensuite présenté trois approches pour la localisation de l'événement fautif. La première consiste à établir une fenêtre d'événements sur laquelle nous travaillons afin de rétablir la cohérence entre les trajectoires du modèle de bon comportement et les observations reçues. Cette méthode est une solution possible à notre problème mais elle présente deux défauts : le nombre de calculs en ligne et le problème de détectabilité. Si un événement n'est pas détectable, alors il est impossible à cette méthode de converger. La deuxième approche proposée consiste à utiliser une approche largement employée dans le domaine du diagnostic des systèmes à événements discrets : l'approche diagnostiqueur. Cette approche permet de ne faire aucun calcul en ligne autre que le suivi du modèle. En outre, le problème de détectabilité qui se retrouve dans le diagnostiqueur n'empêche pas sa construction. Cependant, la construction du diagnostiqueur passe par le produit synchronisé des automates, ce qui pose un problème d'explosion du nombre d'états. Pour contourner ce problème, nous avons développé une approche similaire mais en nous basant sur le formalisme des réseaux de Petri afin d'exploiter leur pouvoir de représentation. L'approche diagnostiqueur fournit un ensemble d'hypothèses sur l'insertion ou l'absence de tous les événements qui, par leur insertion ou leur absence, expliquent les observations reçues.

Le résultat de cette thèse a été le développement d'un programme qui permet de construire de façon automatique les diagnostiqueurs à partir d'un réseau de Petri de bon comportement. Pour valider cette approche, une plateforme d'applications a été également développée. Elle

permet la surveillance d'un réseau de communication, la détection d'une incohérence entre les événements attendus et ceux reçus, et la visualisation sur une interface graphique de l'évolution d'un réseau de Petri.

La détection et la localisation de défaillances intermittentes est un problème complexe et reste très peu étudiées. Même si nous avons apporté une première solution dans le cas de l'insertion ou l'absence d'un événement. Il reste encore de nombreux points à éclaircir et de nombreuses perspectives à developper.

8.2 Discussions et perspectives

Extension de l'approche à moyen terme

Dans notre approche, l'une de nos hypothèses les plus fortes est la supposition de l'insertion ou de l'absence d'un unique événement. Une étude doit être menée pour écarter cette hypothèse. En effet, nous avons été confrontés à ce problème lors de la validation sur l'application. La souplesse dans la construction des modèles de bon comportement nous a permis de contourner ce problème mais cela n'est pas entièrement satisfaisant. Nous avons montré une possible utilisation de modèles de classe de fautes pour atteindre cet objectif. Mais il est nécessaire d'approfondir les méthodes développées pour concevoir une approche qui considère des séquences d'événements.

Une autre hypothèse suppose aussi que la séquence d'événements reçus divergera du comportement normal et donc que la défaillance pourra être détectée. Or, si par exemple, la défaillance entraîne l'émission d'une demande de sortie de la rampe, le système réagira à cette demande en sortant la rampe et aucune détection ne sera faite. Cependant, l'utilisateur verra néanmoins un dysfonctionnement au niveau de la rampe qu'il signalera. L'absence de détection est, en fait, déjà une information. En effet, nous savons qu'entre le moment où la demande est émise et celui où le moteur de la rampe est mis en marche, il n'y a pas de problème a priori. Seuls les événements correspondant à la demande et à la mise en marche sont suspects. Une approche pourrait donc être développée pour prendre en compte les symptômes au niveau fonctionnel (vu par l'utilisateur) afin d'apporter un diagnostic plus précis.

Nous avons tout au long de cette thèse traité d'événements observables. Or, il se peut que l'événement qui s'insère ne fasse pas partie du modèle de bon comportement tout en induisant un mauvais comportement de notre fonction. Ce genre d'événements n'étant pas pris en compte, il est impossible dans notre approche de les détecter et par conséquent de les localiser. Par exemple, une saturation du réseau de communication peut empêcher l'émission de message.

De la même manière, si l'événement qui s'insère n'est pas observable, nous ne suspectons pas cet événement. Par exemple, il existe des capteurs qui sont reliés directement à des calculateurs maîtres et dont les informations ne sont pas communiquées aux autres calculateurs. Ce sont donc des informations qui ne sont pas disponibles pour notre approche. La question, qui se pose ici, concerne finalement le problème de la projection de notre modèle sur nos observables et par conséquent, de la fusion de certains états liés par des événements non ob-

servables. Nous retrouvons le problème de détectabilité/diagnosticabilité qui apparaît dans la littérature scientifique. Cependant, l'ensemble des variables internes (donc non observables) peut être suspecté. Cet ensemble est néanmoins réduit dans la mesure où seules les variables internes des états fusionnés dans lesquels le modèle se trouve lors de la détection sont suspectées. Une étude peut donc être menée pour réduire encore cet ensemble. Finalement, il est nécessaire de se pencher sur ces événements non observables et voir dans quelle mesure ils peuvent influencer les conclusions que nous avons données.

En outre, lors de la projection sur les observables, certaines variables observables peuvent être associées à des variables non observables. Il faut là encore prendre en compte ces variables dans l'ensemble des variables suspectes. Une approche doit donc être développée pour étudier le système en prenant en compte les variables observables et non observables. Cependant, la tendance actuelle est de développer de nouveaux capteurs intelligents qui se connectent directement sur les réseaux de communication. Cela nous permettra d'avoir plus de variables observables et, par conséquent, d'améliorer le diagnostic du système.

Les hypothèses faites au cours de cette thèse doivent être peu à peu écartées pour assurer un diagnostic précis et fiable. Cependant, il faut rappeler que le cas de défaillances fugitives et intermittentes est difficile dans la mesure où peu de connaissances existent sur ces défaillances. L'objectif est de réduire la taille du système à analyser et à diagnostiquer.

D'autres axes de recherches peuvent être explorées pour améliorer le diagnostic comme les contraintes temporelles, les analyses de détectabilité et de diagnosticabilité...

Axes de recherches parallèles

Contraintes temporelles

Les contraintes temporelles n'ont pas été prises en compte dans nos travaux. Il est intéressant de considérer ces aspects car ils permettent de diagnostiquer certaines défaillances. Par exemple, la réception de l'événement en dehors des contraintes temporelles peut être le résultat d'une défaillance. De nombreuses approches concernant les aspects temporels ont été étudiées dans la littérature scientifique dont il a été donné un bref aperçu dans le chapitre 3, avec notamment, l'utilisation de chroniques.

Détectabilité

Dans le chapitre 5, le problème de détectabilité a été mentioné car il empêchait la convergence de l'algorithme. Dans l'approche diagnostiqueur, il n'y a pas de problème de convergence mais il existe toujours ce problème de détectabilité. Il peut s'énoncer de la manière suivante : s'il existe un marquage dans le diagnostiqueur dont le label contient deux couples de labels (M, Ok) et $(M, Ft_i{}^+)$ (réciproquement $(M, Ft_i{}^-)$) alors la détection de l'insertion de l'évenement associé à t_i (réciproquement de l'absence de l'événement associé à t_i) n'est pas possible. En effet, si à un même marquage est associée une hypothèse de fautes et une hypothèse de bon comportement alors la réception des événements suivants ne permettra pas de discriminer les deux hypothèses. Les trajectoires du modèle qui suivent sont identiques, car les marquages sont identiques. Une étude peut donc être menée pour déterminer quels

sont les événements qui peuvent être détectés.

Diagnosticabilité

Une fois détectées, les fautes doivent être diagnostiquées, ce qui n'est pas toujours possible. En effet, il existe de nombreux états où les labels associés contiennent plusieurs hypothèses de fautes. Nous avons, dans notre approche, stoppé la construction du diagnostiqueur dès que les labels associés aux états ne correspondent plus qu'à des fautes. Une solution serait de continuer à développer le diagnostiqueur jusqu'à obtenir des états avec un seul label de faute $\{(M, Ft_i{}^+)\}$. De la même manière que pour la détectabilité, si un marquage a un label qui contient deux couples de labels $(M, Ft_j{}^+)$ et $(M, Ft_i{}^+)$ alors les fautes $Ft_i{}^+$ et $Ft_j{}^+$ ne sont pas discriminables. Une étude sur la diagnosticabilité pourrait permettre d'obtenir un diagnostic plus précis.

Bibliographie

A.Debiolles, L. Oukhellou et P. Aknin (2006). Output coding of spatially dependent subclassifiers in evidential framework. application to the diagnosis of railway track/vehicle transmission system. In : *9th International Conference on Information Fusion*. pp. 1–6.

Al-Malki, M.F. et D. Gu (2003). Ann-based fdi with application to high performance helicopter. In : *International Conference on Simulation and Modelling*. Marbella (Espagne).

Al-Malki, M.F. et D. Gu (2006). Advantages of using μ-synthesis for fault tolerant flight control system. In : *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS'2006*. Beijing (Chine). pp. 199–204.

Anderson, R.J. et S.R. Aylward (1993). Lab testing of neural networks for improved aircraft onboard-diagnostics on flight-ready hardware. In : *Proceedings of Reliability and Maintainability Symposium*. pp. 404–410.

Benveniste, A., E. Fabre, C. Jard et S. Haar (2003). Diagnosis of asynchronous discrete event systems : A net unfolding approach. *IEEE Transactions on Automatic Control* **48**(5), 714–727.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Boatas, A. (2001). Diagnostic par estimation de densité conditionnelle. Application à la surveillance du convertisseur catalytique d'une automobile. Thèse de doctorat. Université de Compiègne/PSA.

Boatas, A., M.A. Dilliès-Peltier et B. Dubuisson (2000). Obd using statistical pattern recognition. In : *Proceedings of SAE Future Transportation Technology Conference, OOFTT19*. Costa Mesa, (USA).

Bobi, M.A. Sanz, J. M. Besada Juez, R. Palacios, A. Muñoz, R. García-Escudero, M. Pérez Alonso et A. L. Matesanz (2001). Diagnosis of the electrical motors of a train using self-organised maps. In : *3rd IEEE International Symposium on Diagnostics for Electrical Machines Power Electronics and Drives*.

Breuer, M. A. (1973). Testing for intermittent faults in digital circuits. *IEEE Transactions on Computer* **22**(3), 241–246.

Buchanan, BG et EH Shortliffe (1984). *Rule Based Expert Systems : The Mycin Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley Longman Publishing Co, Inc.

Cabasino, M.P., A. Giua et C. Seatzu (2007). Marking estimation of petri nets with arbitrary transition labeling. In : *1st IFAC Workshop on Dependable Control of Discrete Systems, DCDS'07*. Cachan, Paris (France). pp. 109–114.

Caines, P.E., R. Greiner et S. Wang (1988). Dynamical logic observers for finite automata. In : *Proceedings of the 27th IEEE Conference on Decision and Control, CDC'88*. Austin, Texas (USA). pp. 226–233.

Capriglione, D., C. Liguori, C. Pianese et A. Pietrosanto (2003). On-line sensor fault detection, isolation, and accommodation in automotive engines. *IEEE Transactions on Instrumentation and Measurement* **52**(4), 1182–1189.

Capriglione, D., C. Liguori et A. Pietrosanto (2007). Real-time implementation of ifdia scheme in automotive systems. *IEEE Transactions on Instrumentation and Measurement* **56**(3), 824–830.

Cardoso, J., R. Valette et D. Dubois (1996). Fuzzy petri net : an overview. In : *13th IFAC World Congress*. Vol. J. San Francisco (USA). pp. 443–448.

Cardoso, J., R. Valette et D. Dubois. (1999). Possibilistic petri nets. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics* **19**, 200–206.

Chang, T., J. Kim et J. Lee (2000). On-board dynamics failure detection of the two-motor-driven electric vehicle system. *52nd IEEE Fall Vehicular Technology Conference* **5**, 2047–2053.

Chatain, T. et C. Jard (2004). Symbolic diagnosis of partially observable concurrent systems. In : *FORTE*. pp. 326–342.

Chen, W. et M. Saif (2003). Fault detection and accommodation in nonlinear time-delay systems. *Proceedings of the American Control Conference* **5**, 4255–4260.

Chen, Y.M. et M.L. Lee (2002). Neural networks-based scheme for system failure detection and diagnosis. *Mathematics and Computers in Simulation* **58(2)**, 101–109.

Chingiz, H. et C. Fikret (2005). Sensor and control surface/actuator failure detection and isolation applied to f-16 flight dynamic. *Aircraft Engineering and Aerospace Technology* **7**(2), 152–160.

Chung, S., C. Wu et M. Jeng (2003). Failure diagnosis : a case study on modeling and analysis by petri nets. *Proceedings of IEEE International Conference on Systems, Man and Cybernetics* **3**, 2727–2732.

Combacau, M. (1991). Commande et surveillance des systèmes à événements discrets complexes, application aux ateliers flexibles. Thèse de doctorat. Université Paul Sabatier de Toulouse.

Console, L. et G Friedrich (1994). Introduction. *Annals of Mathematics and Artificial Intelligence*.

Console, L. et P. Torasso (1990). Integrating models of the correct behavior into abductive diagnosis. In : *ECAI*. pp. 160–166.

Console, L. et P. Torasso (1991). A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence* **7**, 133–141.

Contant, O., S. Lafortune et D. Teneketzis (2002). Failure diagnosis of discrete event systems : The case of intermittent faults. In : *Proceedings of the 41st IEEE Conference on Decision and Control, CDC'02*. Las Vegas (USA). pp. 4006–4011.

Contant, O., S. Lafortune et D. Teneketzis (2004). Diagnosis of intermittent faults. *Discrete Event Dynamic Systems* **14**, 171–202(32).

Cordier, M.O., X. Le Guillou, S. Robin, L. Rozé et T. Vidal (2007). Distributed chronicle for on-line diagnosis of web services. In : *Proceedings of the 18th International Workshop on Principles of Diagnosis, DX'07*. Nashville, Tenessee (USA). pp. 37–44.

Correcher, A., E. Garcia, F. Morant, E. Quiles et R. Blasco-Gimenez (2003). Intermittent failure diagnosis in industrial processes. In : *IEEE International Symposium on Industrial Electronics, ISIE '03*. Vol. 2. pp. 4006–4011.

Crossman, J., H. Guo, Y. Lu et J. Cardillo (2003). Automotive fault diagnosis (part i) : Signal fault analysis, feature extraction and quasi optimal feature selection. *IEEE Transaction on Vehicular* **52**(4), 1063–1075.

Davis, R. et W. Hamscher (1988). Model-based reasoning : troubleshooting. pp. 297–346.

de Freitas, N. (2002). Rao-blackwellised particle filtering for fault diagnosis. *Proceedings of IEEE Aerospace Conference* **4**, 1767–1772.

de Kleer, J. (1991). Focusing on probable diagnoses. In : *AAAI*. pp. 842–848.

de Kleer, J., A. K. Mackworth et R. Reiter (1992). Characterizing diagnoses and systems. *Artificial Intelligence* **56**(2-3), 197–222.

de Kleer, J. et B. C. Williams (1989). Diagnosis with behavioral modes. In : *IJCAI*. pp. 1324–1330.

de Kleer, Johan (1990). Using crude probability estimates to guide diagnosis. *Artif. Intell.* **45**(3), 381–391.

Debiolles, A., Th. Denoeux, L. Oukhellou et P. Aknin (2004). Combined use of partial least squares regression and neural network for diagnosis tasks. In : *17th International Conference on Pattern Recognition 2004, ICPR'04*. pp. 573–576.

Debouk, R., S. Lafortune et D. Teneketzis (1998). A coordinated decentralized protocol for failure diagnosis of discrete event systems. In : *Proceedings of the Workshop on Discrete Event Systems (WODES'98)*. pp. 138–143.

Diaz, M. (2001). *Les réseaux de Petri*. Hermes Science Europe Ltd.

Dousson, C. et P. Le Maigat (2006). Improvement of chronicle-based monitoring using temporal focalization and hierarchization. In : *Proceedings of the 17th International Workshop on Principles of Diagnosis, DX'06*. Burgos (Espagne). pp. 77–84.

Dousson, Christophe, Paul Gaborit et Malik Ghallab (1993). Situation recognition : Representation and algorithms. In : *IJCAI*. pp. 166–174.

Dubuisson, B. (2001). *Diagnostic, intelligence artificielle et reconnaissance des formes*. Hermes Science Europe Ltd.

Dupont, P. et L. Miclet (1998). Inférence grammaticale régulière : fondements théoriques et principaux algorithmes. Technical Report 3449. INRIA.

Emerson, E.A. (1990). Temporal and modal logic. pp. 995–1072.

Faure, P.P. (2001). Une approche à base de modèles fondés sur les intervalles pour la génération automatique d'arbres de diagnostic optimaux : application au domaine de l'automobile. Thèse de doctorat. LAAS-CNRS.

Fekih, A., H. Xu et F. N. Chowdhury (2006). Two neural net-learning methods for model based fault detection. In : *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS'2006*. Beijing (Chine). pp. 85–90.

Fellouah, R., W.C. Lu, F. Mora-Camino et A. Doncescu (2006). Differential flatness and fault detection in flight guidance dynamics. In : *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS'2006*. Beijing (Chine). pp. 211–216.

Ganguli, R. (2003). Application of fuzzy logic for fault isolation of jet engines. *Journal of Engineering for Gas Turbines and Power (Transactions of the ASME)* **123**(3), 617–623.

Gelgele, H.L. et K. Wang (1998). An expert system for engine fault diagnosis : development and application. *Journal of Intelligent Manufacturing* **9**, 539–545(7).

Genc, S. et S. Lafortune (2003). Distributed diagnosis of discrete-event systems using Petri nets. In : *ICATPN'03*. Eindhoven (The Netherlands). pp. 316–336.

Genc, S. et S. Lafortune (2005). A distributed algorithm for on-line diagnosis of place-bordered petri nets. In : *IFAC World Congress'05*. Pragues (République Tchèque).

Gertler, J., Ed.) (1998). *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker.

Ghazel, M., A. Toguyéni et M. Bigand (2005). A monitoring approach for discrete event systems based on a time petri net model. In : *IFAC World Congress'05*. Pragues (République Tchèque).

Girard, Jean-Yves (1987). Linear logic. *Theoretical Computer Science* **50**, 1–102.

Giua, A. (1997). Petri net states estimators based on event observation. In : *Proceedings of the 36th IEEE Conference on Decision and Control, CDC'97*. Vol. 4. San Diego, California (USA). pp. 4086–4091.

Giua, A. et C. Seatzu (2002). Observability of place/transition nets. *IEEE Transactions on Automatic Control* **47**(9), 1424–1437.

Gomez, G., G. Campion, M. Gevres et P. Willem (2000). A case study of physical diagnosis for aircraft engines. *Proceedings of the American Control Conference 2000* **4**, 2383–2387.

Grastien, A. (2005). Diagnostic décentralisé et en-ligne de systèmes à événements discrets reconfigurables. Thèse de doctorat. Université de Rennes I.

Hadjicostis, C.N. et G.C. Verghese (1999). Monitoring discrete event systems using Petri net embeddings. In : *ICATPN'99*. Virginie (USA). pp. 188–207.

Hamscher, W., Console, L. et de Kleer, J., Eds.) (1992). *Readings in model-based diagnosis*. Morgan Kaufmann Publishers Inc.

Harel, D. (1987). Statecharts : A visual formalism for complex systems. *Science of Computer Programming* **8**, 231–274.

Holloway, L.E. et B.H. Krogh (1990). Fault detection and diagnosis in manufacturing systems : a behavioral model approach. *Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing* pp. 252–259.

Hutter, F. et R. Dearden (2003). Efficient on-line fault diagnosis for nonlinear systems. In : *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Hwang, I., J. Hwang et C.J. Tomlin (2003). Flight-mode-based aircraft conflict detection using a residual-mean interacting multiple model algorithm. In : *AIAA Guidance, Navigation and Control Conference*.

Isermann, R. (1997). Supervision, fault-detection and fault-diagnosis methods - an introduction. *Control Engineering Practice* **5**, 639–652(14).

Jakubek, S. et T. Strasser (2002). Neural networks applied to automatic fault detection. *The 45th Midwest Symposium on Circuits and Systems (MWSCAS'02)* **1**, 639–642.

Jard, J.C., T. Chatain et P. Bourhis (2005). Diagnostic temporel dans les systèmes répartis à l'aide de dépliages de réseaux de petri temporels. **39**, 351–365.

Jensen, H.C.B et R. Wisniewski (2002). Fault detection and isolation for spacecraft : Geometric approach. In : *Proceedings of AIAA Guidance, Navigation and Control*. Monterey, California (USA).

Jiang, S. et R. Kumar (2006). Diagnosis of repeated failures for discrete event systems with linear-time temporal-logic specifications. In : *IEEE Transactions on Automation Science and Engineering*. Vol. 3. pp. 47–59.

Jiang, S., R. Kumar et H.E. Garcia (2003). Diagnosis of repeated/intermittent failures in discrete event systems. In : *Proceedings of the IEEE International Conference on Robotics and Automation*. Vol. 19 :2. pp. 310–323.

Jiroveanu, G. et R.K. Boel (2005*a*). Distributed diagnosis for petri nets models with unobservable interactions via common places. *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference, CDC-ECC'05* pp. 6305–6310.

Jiroveanu, G. et R.K. Boel (2005*b*). Petri net model-based distributed diagnosis for large interacting systems. In : *Proceedings of the 16th International Workshop on Principles of Diagnosis, DX'05*. Monterey, Californie (USA).

Julliand, J. (2003). Automates finis et applications, cours de dea i.a.p., tronc commun.

Jéron, T., H. Marchand, S. Pinchinat et M.-O Cordier (2006). Supervision patterns in discrete event systems diagnosis. In : *Proceedings of the Workshop on Discrete Event Systems (WODES'06)*. pp. 262–268.

Kamal, S. et C. V. Page (1974). Intermittent faults : A model and a detection procedure. *IEEE Transactions on Computers* **23**(7), 713–719.

Karp, R. M. et R. E. Miller (1969). Parallel program schemata. *Journal of Computer and System Sciences* **3**(2), 147–195.

Kashi, K., D. Nissing, D. Kesselgruber et D. Söffker (2006). Fault diagnosis of an active suspension control system. In : *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS'2006*. Beijing (Chine). pp. 535–540.

Kimmich, F., A. Schwarte et R. Isermann (2005). Fault detection for modern diesel engines using signal- and process model-based methods. *Control Engineering Practice* **13**(2), 189–203.

Lefebvre, D. et C. Delherm (2005). Diagnosis with causality relationships and directed paths in Petri net models. In : *IFAC World Congress'05*. Pragues (République Tchèque).

Li, P. et R. M. Goodall (2003). Model based approach to railway vehicle fault detection and isolation. In : *Electronics Systems and Control Research Division, 1st Divisional Mini-conference*.

Li, P. et R. M. Goodall (2004). Model-based condition monitoring for railway vehicle systems. In : *Proceedings of the Control 2004 Conference*.

Li, P., R. Goodall, P. Weston, C. Seng Ling, C. Goodman et C. Roberts (2007). Estimation of railway vehicle suspension parameters for condition monitoring. *Control Engineering Practice* **15(1)**, 43–55.

Lopez-Varela, C. (2007). Détection et diagnostic basés cohérence pour les systèmes à événements discrets : vers la prise en compte des erreurs de modélisation. Thèse de doctorat. Université de Toulouse.

Lu, Y., J. Crossman, Z-H. Chen et J. Cardillo (2003). Automotive fault diagnosis (part ii) : A distributed agent diagnostic system. *IEEE Transaction on Vehicular* **52**(4), 1076–1098.

Madden, M.G.M. et P.J. Nolan (1999). Monitoring and diagnosis of multiple incipient faults using fault tree induction. *IEEE Proceedings - Control Theory and Applications* **146**(2), 204–212.

Merlin, P. (1974). A study of the recoverability of computing systems. Thèse de doctorat. Université de Californie, Irvine (USA).

Mirabadi, A., N. Mort et F. Schmid (1998). Fault detection and isolation in multisensor train navigation systems. In : *International Conference on Control*. Vol. 2. pp. 969–974.

Mouchaweh, S., A. Philippot, V. Carré-Ménétrier et B. Riera (2006). Timed-event-state-based diagnoser for manufacturing systems. In : *7th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services*. Niagara Falls, Ontario, Canada.

Mounir-Alaoui, A. (1990). Raisonnement temporel pour la planification et la reconnaissance de situations. Thèse de doctorat. Université Paul Sabatier de Toulouse.

Murphey, Y.L. (2002). Intelligent signal segment fault detection using fuzzy logic. *Proceedings of the IEEE International Conference on Fuzzy Systems* **1**, 12–17.

Murphy, R. R. et D. Hershberger (1996). Classifying and recovering from sensing failures in autonomous mobile robots. In : *Proceedings of the 13th National Conference on Artificial Intelligence*. Vol. 2. pp. 922–929.

Nitsche, C., S. Schroedl et W. Weiss (2004). Onboard diagnostics concept for fuel cell vehicles using adaptive modelling. *IEEE Intelligent Vehicles Symposium* pp. 127–132.

Nyberg, M. (2002). Criterions for detectability and strong detectability of faults in linear systems. *International Journal of Control* **75**, 490–501(12).

Olive, X. (2003). Approche intégrée à base de modèles pour le diagnostic hors ligne et la conception : application au domaine de l'automobile. Thèse de doctorat. LAAS-CNRS.

Papadopoulos, Y. (2003). Model-based system monitoring and diagnosis of failures using statecharts and fault trees. *Reliability Engineering and System Safety* **81(3)**, 325–341.

Papadopoulos, Y. et J. MacDermid (2001). Automated safety monitoring : A review and classification of methods. *International Journal of Condition Monitoring and Diagnostic Engineering Management* **4(4)**, 14–32.

Patton, R. J., J. Chen et C. J. Lopez-Toribio (1998). Fuzzy observers for non-linear dynamic systems fault diagnosis. In : *Proceedings of the 37th IEEE Conference on Decision and Control, CDC'98*. pp. 84–89.

Pencolé, Y. (2002). Diagnostic décentralisé de systèmes à événements discrets : Application aux réseaux de télécommunications. Thèse de doctorat. UNIVERSITE RENNES I.

Philippot, A. (2006). Contribution au diagnostic décentralisé des SED : application aux systèmes manufacturiers. Thèse de doctorat. Université de Reims Champagne-Ardenne, France.

Philippot, A., S. Mouchaweh et V. Carré-Ménétrier (2007). Unconditional decentralized structure for the fault diagnosis of discrete event systems. In : *1st IFAC Workshop on Dependable Control of Discrete-event Systems, DCDS'07*. Cachan, France.

Poulard, H. (1996). Statistiques et réseaux de neurones pour un système de diagnostic : application au diagnostic de pannes automobiles. Thèse de doctorat. LAAS-CNRS.

Pradin-Chezalviel, B. et R. Valette (1993). Petri nets and linear logic for process oriented diagnosis. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics* **2**, 264–269.

Qiu, W. et R. Kumar (2007). A protocol for distributed state estimation in discrete event systems. In : *1st IFAC Workshop on Dependable Control of Discrete Systems, DCDS'07*. Cachan, Paris (France). pp. 97–102.

Recht, J.L. (1966). failure mode and effect. *National Safety Council*.

Resencourt, H. (2006). Approche multi-modèle et hiérarchique pour le diagnostic hors-ligne des systèmes embarqués ; application à l'automobile. 7ème Congrès des Doctorants de l'Ecole Doctorale Systèmes (EDSYS).

Ressencourt, H., L. Travé-Massuyès et J. Thomas (2006). Hierarchical modelling and diagnosis for embedded systems. In : *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS'2006)*. Beijing (Chine). pp. 553–558. Également dans *17th International Workshop on Principles of Diagnosis (DX'06)*, Burgos, Espagne, 2006, pages 235–242.

Reymonet, A., J. Thomas et N. Aussenac-Gilles (2007*a*). Modélisation de Ressources Termino-Ontologiques en OWL. In : *Actes des 18e journées francophones d'ingénierie des connaissances*.

Reymonet, A., J. Thomas et N. Aussenac-Gilles (2007*b*). Modelling ontological and terminological resources in owl dl. In : *Proceedings of ISWC '07 workshop "From Text to Knowledge : The Lexicon/Ontology Interface" (OntoLex '07)*.

Reymonet, A., N. Aussenac-Gilles et J. Thomas (2006). Tâche, domaine et application : influences sur le processus de modélisation de connaissances. In : *Actes des 17e journées francophones d'ingénierie des connaissances*.

Ripley, B. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rosembrock, H.H. (1970). *State-space and multivariable theory*. John Wiley and Sons.

Roumeliotis, S., G. Sukhatme et G. Bekey (1998). Sensor fault detection and identification in a mobile robot. Vol. 3. Victoria, BC (Canada). pp. 1383–1388.

Rozé, L. et M.-O. Cordier (2002). Diagnosing discrete-event systems : extending the "diagnoser approach" to deal with telecommunication networks. *Journal on Discrete-Event Dynamic Systems : Theory and Applications (JDEDS)* **12**(1), 43–81.

Rozé, L. et P. Laborie (1998). Supervision of telecommunication network : Extending the diagnoser approach. In : *Proceedings of the 9th International Workshop on Principles of Diagnosis, DX'98*. Cape Cod (USA).

Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen et D. Teneketzis (1996). Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology* **4**(2), 105–124.

Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen et D. Teneketzis (Sep 1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control* **40**(9), 1555–1575.

Sampath, M., S. Lafortune et D. Teneketzis (1998). Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control* **43(7)**, 908–929.

Savir, J. (1980). Detection of single intermittent faults in sequential circuits. *IEEE Transactions on Computers* **29**(7), 673–678.

Schwarte, A. et R. Isermann (2002). Neural network applications for model based fault detection with parity equations. In : *Proceedings of the 15th IFAC World Congress*. Barcelone (Espagne).

Sharkey, A.J.C., G.O. Chandroth et N.E. Sharkey (2000). Acoustic emission, cylinder pressure and vibration : A multisensor approach to robust fault diagnosis. In : *International Joint Conference on Neural Networks*. Como (Italie).

Shraim, H., A. Rabhi, M. Ouladsine, N. K. M'Sirdi et L. Fridman (2006). A diagnosis preview for the vehicle basing on the estimations of tire pressure and the effective radius of the wheel. In : *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS'2006*. Beijing (Chine). pp. 559–564.

Skarlatos, D., K. Karakasis et A. Trochidis (2004). Railway wheel fault diagnosis using a fuzzy-logic method. *Applied Acoustics* **65(10)**, 951–966.

Soldani, S., M. Combacau, A. Subias et J. Thomas (2007*a*). Intermittent fault diagnosis : a diagnoser derived from the normal behavior. In : *1st IFAC Workshop on Dependable Control of Discrete Systems, DCDS'07*. Cachan, Paris (France). pp. 261–266. Also in *18th International Workshop on Principles of Diagnosis, DX'07*, pages 391–396, Nashville, Tenessee (USA), (mai 2007).

Soldani, S., M. Combacau, A. Subias et J. Thomas (2007*b*). On-board diagnosis system for intermittent fault : Application in automotive industry. In : *7th IFAC International Conference on Fieldbuses and Networks in industrial and Embedded Systems, FET'07*. Toulouse (France). pp. 151–158.

Soldani, S., M. Combacau, J. Thomas et A. Subias (2006). Intermittent fault detection through message exchanges : a coherence based approach. In : *6th IFAC Symposium on*

Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS'2006. Beijing (Chine). pp. 1549–1554. Also in *17th International Workshop on Principles of Diagnosis, DX'06*, pages 251–256, Burgos (Espagne), (Juin 2006).

Steinbauer, Gerald et Franz Wotawa (2005). Detecting and locating faults in the control software of autonomous mobile robots. In : *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*. Edinburgh, Ecosse (GB). pp. 1742–1743.

Straky, H., M. Munchhof et R. Isermann (2002). A model based supervision system for the hydraulics of passenger car braking systems. In : *Proceedings of the 15th IFAC World Congress*. Barcelone (Espagne).

Struss, P. et O. Dressler (1989). "physical negation" integrating fault models into the general diagnostic engine. In : *IJCAI*. pp. 1318–1323.

Su, R., W. M. Wonham, J. Kurien et X. Koutsoukos (2002). Distributed diagnosis for qualitative systems. In : *Proceedings of the Sixth International Workshop on Discrete Event Systems, WODES'02*. Washington (USA). pp. 169–174.

Tabakow, I. (2007). Using place invariants and test point placement to isolate faults in discrete event systems. *j-jucs* **13**(2), 224–243.

Takagi, T. et M. Sugeno (1985). Fuzzy identification of systems and its applications to modeling and control. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics* **15**, 116–132.

Travé-Massuyès, L., T. Escobet et S. Spanache (2003). Diagnosability analysis based on component supported analytical redundancy relations. In : *5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS'2003*. Washington (USA). pp. 897–902.

Ushio, T., I. Onishi et K. Okuda (1998). Fault detection based on petri net models with faulty behaviors. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics* **1**, 113–118.

Valette, R. et L. Kunzle (1994). Réseaux de petri pour la détection et le diagnostic. In : *Journées Sûreté, surveillance, supervision : détection et localisation des défaillances*.

Venkatasubramanian, V., R. Rengaswamy, K. Yin et S.N. Kavuri (2001). A review of process fault detection and diagnosis part i : Quantitative model-based methods. *Computers and Chemical Engineering* **27(3)**, 293–311.

Venkateswaran, N., M. S. Siva et P. S. Goel (2002). Analytical redundancy based fault detection of gyroscopes in spacecraft applications. *Acta Astronautica* **50**, 535–545.

Verma, V., G. Gordon, R. Simmons et S. Thrun (2004). Real-time fault diagnosis [robot fault diagnosis]. *Robotics & Automation Magazine* **11**(2), 56–66.

Villemeur, A. (1988). *sûreté de fonctionnement des systèmes industriels : fiabilité, facteurs humains, informatisation*. Eyrolles.

Washington, Richard (2000). On-board real-time state and fault identification for rovers. In : *Proceedings of the IEEE International Conference on Robotics and Automation*. Vol. 2. pp. 1175–1181.

Whiteford, T. M. et R. Kwong (2007). Probabilistic fault diagnosis in discrete event systems with incomplete models. In : *1st IFAC Workshop on Dependable Control of Discrete Systems, DCDS'07*. Cachan, Paris (France). pp. 49–54.

Wu, X. et G. Campion (2004). Fault detection and isolation of systems with slowly varying parameters : simulation with a simplified aircraft turbo engine model. *Mechanical Systems and Signal Processing* **18(2)**, 353–366.

Yager, R. et D.P. Filev (1994). *Essentials of fuzzy modeling and control*. Wiley-Interscience. New York, NY, USA.

Yeung, D. et R. Kwong (2005). Fault diagnosis in discrete-event systems : incomplete models and learning. *2005 American Control Conference* **5**, 3327–3332.

Yu, L., D.J. Cleary et P.E. Cuddihy (2004). A novel approach to aircraft engine anomaly detection and diagnostics. *Proceedings of IEEE Aerospace Conference* **5**, 3468–3475.

Zadeh, L.A. (1965). Fuzzy sets. *Information and Control* **8**(3), 338–353.

Zhang, Y. et X.R. Li (1997). Detection and diagnosis of sensor and actuator failures using interacting multiple-model estimator. In : *36th IEEE Conference on Decision and Control, CDC'97*. San Diego, CA (USA). pp. 4475–4480.

Zwingelstein, G. (1995). *Diagnostic des défaillances, théorie et pratique pour les systèmes industriels*. Hermès.

Liste des figures

Siegfried SOLDANI

Vers le diagnostic embarqué de défaillances dans les systèmes à événements discrets : Application au domaine de l'automobile

Directeur de thèse : Michel COMBACAU, Professeur des Universités

Co-encadrants : Audine SUBIAS et Jérôme THOMAS

Spécialité : Systèmes automatiques

Présentée le *12 septembre* au *LAAS/CNRS*

Résumé

Résumé

Cette thèse, réalisée dans le cadre d'une convention CIFRE entre ACTIA et le LAAS/CNRS, porte sur la détection et le diagnostic de défaillances fugitives et intermittentes dans les systèmes embarqués, avec une application dans le domaine automobile.
Les méthodes proposées sont basées sur le modèle à événements discrets représentant le fonctionnement normal du comportement observable du système surveillé. L'étape de détection consiste à comparer la séquence d'événements observables émis par le système surveillé et la séquence d'événements attendus par le modèle. L'étape de localisation donne l'ensemble des événements potentiellement responsables des défauts observés. Cette étape se base sur l'approche "diagnostiqueur" développée par (Sampath *et al.*, 1998), dont le principe repose sur la compilation hors-ligne de l'information de diagnostic dans une structure de données (appelé diagnostiqueur) qui relie efficacement les observations aux fautes lors du diagnostic en ligne.
Cette thèse a abouti au développement de plusieurs applications permettant le diagnostic de défauts fugitifs et intermittents. Les travaux ont été validés sur un banc de test représentant une architecture réseau utilisée dans les véhicules de transports urbains (bus, métro...).

Mots clés

Diagnostic à base de modèle, systèmes à évènements discrets, défauts fugitifs et intermittents, systèmes embarqués, domaine automobile.

Laboratoire d'Analyse et d'Architecture des Systèmes - UPR 8001
7, avenue du Colonel Roche, 31077 Toulouse Cedex 4

Towards an on-board faults diagnosis in discrete event systems : Application in automotive industry

Abstract

This thesis deals with the fault diagnosis in the embedded network systems and especially in the automotive systems. It is focused on the detection and localization of intermittent faults in discrete events systems.

The proposed methods are based on a discrete event model representing the normal behavior of the system. The detection step consists in comparing the flow of observable events emitted by the monitored system and the flow foreseen by the model. A localization mechanism, based on the diagnoser approach (Sampath *et al.*, 1998), points out the set of events potentially responsible for the faults. The aim of this approach is to get the diagnosis information into a data structure (off-line) which efficiently joins the observations to the faults during the on-line diagnosis.

This thesis have lead to the development of many applications for the fault diagnosis. This works have been validated on a test bench representing the embedded network system used in the city transports (bus, subways...).

Key words

Model based diagnosis, discrete event systems, intermittents faults, on-board systems, automotive fields.