Diagnostic à base de modèles des systèmes temporisés et d'une sous-classe de systèmes dynamiques hybrides

Haithem Derbel

Université Joseph Fourier - Grenoble I

No. attribué par la bibliothèque

THESE EN COTUTELLE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER - GRENOBLE I ET L'ÉCOLE NATIONALE DES SCIENCES DE L'INFORMATIQUE - TUNISIE

préparée au Département Automatique de GIPSA-lab à Grenoble et l'UR OASIS en Tunisie

dans le cadre de l'École Doctorale :
Électronique, Électrotechnique, Automatique, Traitement du Signal

présentée et soutenue publiquement

par

Haithem DERBEL

le 18 Décembre 2009

Titre :

Diagnostic à base de modèles des systèmes temporisés et d'une sous-classe de systèmes dynamiques hybrides

Directeurs de thèse :

M. Hassane ALLA	(Professeur, Laboratoire GIPSA-lab, Grenoble.)
M. Néjib BEN HADJ-ALOUANE	(Professeur, UR OASIS, Tunisie.)

JURY :

M. René DAVID	DR Émérite CNRS	Président du jury
M. Hervé GUEGUEN	SUPELEC Rennes	Rapporteur
M. Moncef TAJINA	ENSI - Tunisie	Rapporteur
M. Moez YEDDES	ENSI - Tunisie	Examinateur
M. Hassane ALLA	UJF - Grenoble I	Examinateur
M. Néjib BEN HADJ-ALOUANE	ENSI - Tunisie	Examinateur

A la mémoire de mon père,

à ma mère,

à ma chère Emna

et à tous ceux qui pensent à moi...

Remerciements

Ce mémoire marque une étape importante dans ma vie. Je voudrais remercier ici toutes les personnes qui m'ont accompagné pour accomplir l'un des plus importants projets professionnels de ma vie

Je tiens avant tout à exprimer ma profonde reconnaissance à Monsieur Hassane ALLA et Monsieur Néjib BEN HADJ-ALOUANE, qui ont assuré la direction de cette thèse. Je les remercie pour l'aide scientifique qu'ils m'ont toujours apporté, mais surtout pour leur disponibilité, leur soutien et leurs encouragements. Des remerciements très spéciaux pour Monsieur Moez YEDDES, pour toutes les discussions fructueuses, les critiques constructives et l'aide considérable qui ont contribué au contenu de cette thèse.

Je tiens à remercier Monsieur Moncef TAJINA, Professeur à l'Ecole Nationale des Sciences de l'Informatique de Tunis, d'avoir accepté avec Monsieur Hervé GUEGUEN, Professeur à Supelec de Rennes, d'étudier mes travaux et d'en être les rapporteurs ainsi que pour l'intérêt et l'attention qu'ils ont accordés à cette étude. Je remercie énormément les membres du jury pour les conseils et remarques qui m'ont beaucoup aidé à la finalisation de ce document : René DAVID, président du jury, Directeur de Recherche au CNRS, Moez YEDDES Maitre-Assistant à l'Ecole Nationale des Sciences de l'Informatique de Tunis.

Je suis également reconnaissant au soutien financier apporté par le projet de coopération Franco-Tunisien CMCU. Ce soutien, attribué par le Ministère des affaires étrangères et européennes, a été géré par l'EGIDE, que je tiens également à les remercier respectivement.

Mes remerciements vont à tout le personnel du laboratoire Gipsa-lab et de l'unité de recherche OASIS qui m'ont accueilli durant ces trois années. L'ambiance chaleureuse est propice à un travail efficace.

Je tiens tout particulièrement à remercier tous les doctorants du laboratoire (Amine, Cedric, Andra, Van, Hala, Mohamed, Irfan, Lizeth, Adib, Oumayma, Joumana, Simona, Hu, et les autres ...), pour l'ambiance sympathique qu'ils ont réussi à instaurer.

Je tiens à exprimer mon éternelle gratitude à ma chère mère qui m'a toujours soutenu tout au long de mon cursus. Parce qu'elle trouve dans l'achèvement de ce travail l'aboutissement de ses efforts et ses sacrifices. J'exprime ma gratitude à ma chère Emna qui a

toujours été ma motivation la plus importante et qui a été disponible pour m'encourager à achever ce travail. Je remercie tous les membres de ma famille (Wided, Mohamed, Mourad, mon cousin Sami, Cyrine et mon petit choux Slim) pour tous leurs encouragements.

Je ne pourrais pas terminer sans exprimer un remerciement venant du plus profond du coeur à tous mes amis qui m'ont toujours soutenu pendant mes années d'étude (Adel, Hédi, Amine, Imène, Mehdi, Wissem, Youssef, Dilouma, Yamen, Iyed, Khabeb, Ghassen, Rayène et les autres ...).

Table des matières

Table des figures

Liste des Algorithmes

Chapitre 1

Introduction générale

L'automatisation des installations industrielles vise à augmenter la productivité des systèmes et à réduire les coûts de la maintenance des équipements de production. Cependant, la complexité due à l'automatisation des systèmes de production implique des besoins croissants en termes de disponibilité et de performance. Ainsi, il est nécessaire de disposer d'une fonction permettant le diagnostic des défaillances pouvant affecter le fonctionnement du système. Ceci permet d'envisager des actions correctives pour que ce dernier retourne à son fonctionnement nominal. Un module de diagnostic est nécessaire, non seulement pour améliorer les performances et la productivité des systèmes, mais également pour limiter les conséquences des pannes qui peuvent être catastrophiques sur le plan des biens et des vies humaines.

La fonction de diagnostic consiste à détecter une défaillance, de localiser son origine et de déterminer ses causes. Son principe général consiste à confronter les données relevées au cours du fonctionnement réel du système avec la connaissance dont on dispose sur son fonctionnement normal et anormal. Plusieurs chercheurs ont abordé la thématique de la surveillance industrielle et du diagnostic mettant ainsi en évidence l'intérêt manifesté aussi bien par la communauté du contrôle des systèmes avec des approches issues de l'automatique que par la communauté d'Intelligence Artificielle (IA). Les approches de diagnostic développées durant ces dernières décennies peuvent être classées en deux grandes catégories : les approches avec modèles qui se basent sur l'existence d'un modèle du système à surveiller et les approches sans modèles qui se basent sur l'analyse des variables de surveillance et sur l'expertise humaine. Le principe des méthodes avec modèles repose sur la comparaison du comportement prévu par le modèle avec le comportement réellement observé du système. Tout écart entre ces deux comportements sera synonyme de défaillance. L'utilisation d'un modèle du système pour son diagnostic nécessite souvent sa conformité au critère de diagnosticabilité (Lin et Wonham, 1988; Sampath et al., 1995). Il s'agit de vérifier si chaque défaillance peut être détectée et isolée dans un temps fini après l'occurrence du défaut source de la défaillance. En d'autres termes, ce critère

consiste à déterminer si le modèle du système est suffisamment riche en information pour permettre l'identification de tous les défauts affectant son fonctionnement.

De nombreuses approches de diagnostic à base de modèles ont été proposées dans la littérature. On retrouve ainsi les approches issues de la communauté FDI (Fault Detection and Isolation) (Jones, 1973; Isermann, 1984; Chow et Wilsky, 1984), les approches issues de la communauté des Systèmes à Événements Discrets (SED) (Lin et Wonham, 1994; Sampath et al., 1995; Tripakis, 2002; Zad et al., 2003; Ghazel et al., 2005) et celles issues de la communauté DX (Diagnosis eXpert system) (Bousson et al., 1998).

La notion de diagnostic des SED a été introduite au milieu des années 1990 dans les travaux de Sampath (Sampath et al., 1995, 1996), qui ont inspiré de nombreuses extensions (Debouk et al., 2000; Tripakis, 2002; Silveira, 2003; Xue et al., 2005; Zad et al., 2005; Lunze, 2006; Thorsley et Teneketzis, 2005; Bhowal et al., 2007). Cette approche de référence repose sur la compilation d'un automate appelé diagnostiqueur, à partir d'une machine à états finis modélisant le comportement du système. La dynamique du SED dans cette approche est décrite par une séquence d'événements qui caractérisent les transitions d'états du système. Ainsi, la prise en compte du temps est réalisée uniquement par l'ordre d'occurrence des événements. Cela entraîne une perte considérable d'information, indispensable pour l'identification des défauts. En effet, les dates d'occurrence des événements peuvent jouer un rôle considérable dans la discrimination des défauts. Dans certains cas, les comportements défaillants se manifestent uniquement par un changement des dates d'occurrence des événements observables, ce qui rend inappropriée, l'utilisation d'une démarche de diagnostic fondée sur une abstraction purement discrète de l'évolution du système. Dans certains systèmes, il faut aller au delà de la prise en compte du temps, il est alors nécessaire d'introduire une modélisation mixte comprenant des variables continues et des variables discrètes. De tels systèmes, intégrant à la fois une dynamique continue et une dynamique discrète, sont appelés Systèmes Dynamiques Hybrides (SDH).

Dans ce cadre, de nombreuses solutions ont été proposées dans la littérature permettant le diagnostic des SED temporisés (Tripakis, 2002; Bouyer et Chevalier, 2005; Zad et al., 2005; Ghazel et al., 2005) ou celui des SDH (Feng et al., 2000; Koutsoukos et al., 2001; Karsai et al., 2003; Domlan et al., 2004; Bhowal et al., 2007). Des extensions de l'approche de Sampath se sont intéressées à la compilation hors-ligne du diagnostiqueur à partir de modèles SED à temps discret (Zad et al., 2005) ou de modèles SDH à temps discret (Bhowal et al., 2007). Ces modèles représentent la progression du temps moyennant un événement spécial, appelé *tick* d'horloge. Ces approches sont exposées au problème de l'explosion combinatoire due à la représentation discrète de l'information temporelle. Dans (Tripakis, 2002), un diagnostiqueur sous la forme d'un algorithme d'estimation d'état en-ligne, à partir d'un modèle automate temporisé du système, est utilisé pour identifier les défauts qui affectent son fonctionnement.

Contributions de la thèse

Les travaux de ce mémoire portent sur la proposition d'une approche de diagnostic pour les systèmes temporisés les systèmes dynamiques hybrides. Dans un premier axe, nous proposons une démarche de diagnostic pour les systèmes temporisés (Derbel et al., 2006, 2009c). Cette solution repose sur l'utilisation d'un diagnostiqueur compilé hors-ligne, construit à partir d'un modèle automate temporisé du système. Étant donné que ce problème est indécidable dans le cas général d'automates temporisés (Bouyer et Chevalier, 2005), nous considérons une sous-classe d'automates temporisés, caractérisée par un ensemble d'hypothèses. L'algorithme de synthèse du diagnostiqueur que nous proposons repose sur une représentation symbolique de l'espace d'état infini d'un automate temporisé. Une étude de la diagnosticabilité du modèle du système à diagnostiquer ainsi qu'une méthode systématique permettant sa vérification sont présentées.

Dans un deuxième axe, nous proposons une démarche de diagnostic pour une sous-classe de systèmes hybrides dynamiques (Derbel et al., 2009b,d,a). Pour faire face aux complexités que pose l'analyse de la dynamique continue de ces systèmes, nous nous focalisons sur l'étude d'une sous-classe de systèmes dynamiques hybrides modélisés par des automates hybrides rectangulaires vérifiant certaines conditions. Ce modèle permet d'approximer les comportements de la dynamique continue du système à travers l'utilisation de conditions de flux rectangulaires de la forme $\dot{x} \in [a, b]$. Ainsi, notre solution de diagnostic repose sur l'utilisation d'un diagnostiqueur sous la forme d'un algorithme d'estimation d'état en-ligne. Cet algorithme estime à la volée l'état du système ainsi que les défaillances affectant son comportement, à partir de son modèle automate hybride rectangulaire et des observations qu'il génère. Le choix d'une solution en-ligne découle également des nombreuses indécidabilités relatives à ce formalisme hybride. Enfin, nous proposons une notion de diagnosticabilité permettant de s'assurer de l'identification des défauts dans un horizon de temps limité. Une méthode systématique pour la vérification de cette notion, basée sur l'application d'une procédure d'analyse d'accessibilité, est proposée.

Ce mémoire est organisé en quatre chapitres.

Le premier chapitre permet d'exposer les différents concepts liés à notre travail. D'abord, il permet de définir les systèmes concernés par notre approche de diagnostic à savoir, les systèmes automatisés de production. Ensuite, un rappel de la terminologie du diagnostic rencontrée dans la littérature et employée dans ce mémoire est présenté. Enfin, une classification des principales méthodes de diagnostic existantes dans la littérature est proposée.

Dans le deuxième chapitre, nous présentons un état de l'art sur les principales approches de diagnostic à base de modèles, liées à notre problématique de modélisation et de surveillance des SED et des SDH. Ce chapitre nous permet de positionner notre

contribution par rapport aux travaux existants dans la littérature. Dans un premier volet, nous rappelons le cadre de modélisation des SED et des SDH. Nous présentons les caractéristiques de chacun de ces systèmes ainsi que quelques outils permettant leur modélisation. Dans un deuxième volet, nous présentons un survol des principales méthodes de diagnostic des SED et de SDH évoquées dans la littérature.

Le troisième chapitre présente notre démarche de diagnostic pour les systèmes temporisés. Il expose, autour d'un exemple, les différentes étapes nécessaires pour la synthèse en-ligne du diagnostiqueur à partir d'un modèle automate temporisé du système (Derbel et al., 2009c). Nous commençons par présenter formellement le modèle automate temporisé, ses outils d'analyse ainsi que les différentes hypothèses retenues pour permettre la construction du diagnostiqueur. Ensuite, nous détaillons l'algorithme employé pour la synthèse hors-ligne du diagnostiqueur à partir du modèle considéré. Enfin, nous présentons une notion de diagnosticabilité pour le modèle considéré. Une méthode systématique de vérification de cette notion, reposant sur la détection de cycles spéciaux dans le modèle diagnostiqueur, est proposée.

Le dernier chapitre est consacré à la présentation de notre approche de diagnostic d'une sous-classe de systèmes dynamiques hybrides (Derbel et al., 2009b,d). D'abord, nous rappelons le cadre de modélisation considéré dans notre contribution : les automates hybrides rectangulaires, ainsi les méthodes permettant l'analyse comportementale de ce modèle. Nous détaillons par la suite notre démarche de diagnostic de ces systèmes. Cette démarche commence par caractériser les différentes hypothèses de modélisation permettant l'application de notre méthode de diagnostic et la vérification de diagnosticabilité. Ensuite, nous exposons intuitivement puis formellement notre procédure de diagnostic en-ligne. Une notion de diagnosticabilité pour la sous-classe de modèles considérée est définie. Nous proposons enfin une méthode permettant la vérification systématique de cette diagnosticabilité.

Chapitre 2

Diagnostic des défauts dans les systèmes automatisés

2.1 Introduction

Durant ces dernières décennies, l'automatisation des systèmes industriels vise à augmenter les performances de production ainsi que la qualité du produit à travers sa traçabilité et la diminution des coûts de sa fabrication. Dans ce contexte, les systèmes de surveillance des équipements industriels jouent un rôle important pour maintenir la disponibilité des machines et les lignes de production.

Un système de surveillance observe en continu l'évolution des équipements à travers des données quantifiables et/ou qualifiables collectées à partir du système surveillé. Ces données permettent de signaler au bon moment à l'opérateur les écarts détectés par rapport au comportement nominal prévu. Ceci permettra de mettre en oeuvre les actions préventives et correctives. Plusieurs chercheurs ont abordé la thématique de la surveillance industrielle mettant ainsi en évidence l'intérêt manifesté par la communauté scientifique ainsi que les industriels par rapport à cette problématique. Les approches de surveillance sont généralement divisées en deux catégories : les approches de surveillance avec modèles et les approches de surveillance sans modèles. Les premières se basent sur l'existence d'un modèle formel de l'équipement et utilisent généralement les techniques de l'automatique (Combacau, 1991). Dans la deuxième catégorie de méthodologies, le modèle du procédé est inexistant ou difficile à obtenir, et elles se basent ainsi sur des techniques statistiques ou issues du domaine de l'Intelligence Artificielle (IA).

L'objectif de ce chapitre est de présenter les concepts fondamentaux liés au diagnostic et à la surveillance des systèmes automatisés. Dans un premier temps, nous présentons le contexte de notre étude à savoir, les systèmes automatisés de production. Dans la littérature de la surveillance et du diagnostic, on peut trouver plusieurs définitions quelquefois

divergentes. C'est la raison pour laquelle nous donnons les définitions des mots clés nécessaires pour la compréhension de ce rapport. Enfin, nous présentons une classification non exhaustive des méthodes de diagnostic des défauts rencontrées dans la littérature.

2.2 Les Systèmes Automatisés de Production

Les besoins croissants en termes de productivité, de qualité et de disponibilité des systèmes industriels, durant ces dernières décennies, ont fortement participé au développement du processus d'automatisation des systèmes industriels. Cette automatisation vise à augmenter les cadences de production tout en réduisant le temps de maintenance et gardant une certaine flexibilité pour pouvoir s'adapter à la production des nouveaux produits.

2.2.1 Présentation générale

D'une manière générale, un système est un ensemble d'éléments en interaction organisé dans un environnement avec lequel il interagit pour réaliser une fonction qui lui est attribuée (Rosnay, 1975). Un *Système Automatisé de Production* (SAP) permet de remplir, de manière automatique, des fonctions, répondant à certains besoins spécifiques, précédemment assurées par l'homme. Comme il est illustré dans la figure 2.1, inspirée de (Perrin et al., 2004), un SAP est composé d'une Partie Commande (PC) et d'une Partie Opérative (PO).

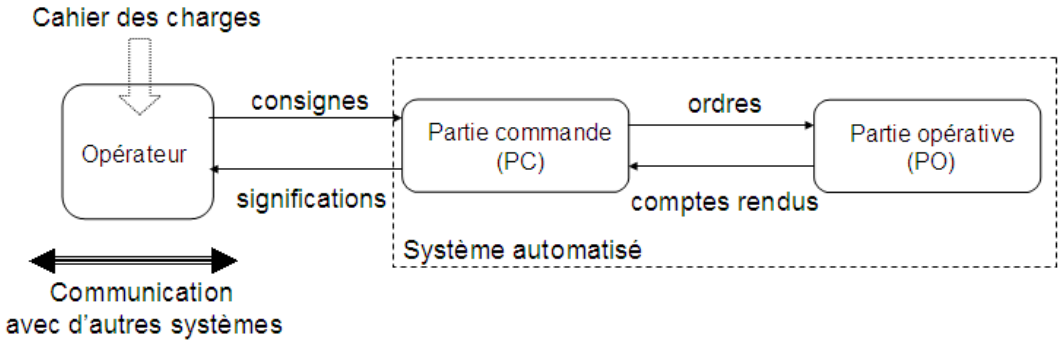

Fig. 2.1 – *Structure d'un SAP.*

- La partie commande n'est autre que la logique de fonctionnement du processus que l'on veut automatiser. Elle envoie des ordres à la PO et permet la communication avec l'opérateur. Les deux composantes principales de cette partie concernent la supervision (le pilotage) et la surveillance (le suivi). La fonction de surveillance

permet de vérifier le bon fonctionnement d'un équipement ou le bon déroulement d'un procédé.
- La partie opérative exécute les ordres envoyés par la PC à l'aide d'actionneurs et lui communique les informations collectées à partir de capteurs, autrement dit les compte-rendus.

2.2.2 Différents types de SAP

Les SAP peuvent être caractérisés à travers la dynamique exprimant leur fonctionnement. Selon l'objectif considéré, il y a trois abstractions possibles pour modéliser cette dynamique : les systèmes continus, les systèmes à événements discrets et les systèmes dynamiques hybrides (Dousson, 2007).

Les systèmes continus :

Les systèmes continus sont constitués d'éléments caractérisés par une ou plusieurs mesures qui peuvent prendre des valeurs réelles lorsque le temps évolue. Les grandeurs peuvent être, par exemple, une position, une vitesse, une accélération, un niveau, une pression, une température, un débit, une tension, etc. La gestion de ces systèmes fait appel à des outils mathématiques aptes à la représentation de la dynamique continue comme les équations différentielles. Les systèmes continus sont perçus par l'automaticien à travers une représentation reposant le plus souvent sur des variables d'état continues et une variable temporelle, continue ou discrète.

Les Systèmes à Événements Discrets :

Un Système à Événements Discrets (SED) est un système à espace d'état discret dont les transitions entre les états sont associées à l'occurrence d'événements discrets asynchrones (Cassandras et Lafortune, 1999). Ces systèmes recouvrent un grand nombre de situations, allant de la circulation de véhicules (en réseau urbain ou en atelier de fabrication) au fonctionnement de machines dans un atelier flexible. Plusieurs modèles mathématiques ont été proposés pour l'analyse et la commande de ces systèmes, en particulier les automates à états finis et les Réseaux de Petri (RdP). Certains modèles de SED, tels que les automates temporisés, utilisent des variables temporelles de nature symbolique, où le temps sert, essentiellement, à définir une chronologie entre les événements.

Les Systèmes Dynamiques Hybrides :

Les Systèmes Dynamiques Hybrides (SDH) couvrent simultanément les deux aspects

continu et discret. Ces systèmes évoluent dans le temps et combinent des variables continues et des variables discrètes (Ben Hadj-Alouane et al., 2006). Un état discret du système peut être vu comme un système continu avec des variables continues reliées par des contraintes. Cependant, la portée de ces contraintes est restreinte à l'état en question. La transition du système d'un état à un autre fait changer son mode de fonctionnement en lui faisant subir d'autres lois continues propres au nouvel état.

2.2.3 Les potentiels dysfonctionnements

L'utilisation des SAP dans le milieu industriel doit répondre à des objectifs prédéfinis. Malheureusement, cette utilisation peut être confrontée à des dysfonctionnements non prévus dont les conséquences sont désastreuses pour la sécurité des hommes et des équipements. Les dysfonctionnements peuvent être de deux types : **externes** ou **internes** (Combacau et al., 2002; Deschamps, 2007).

Les dysfonctionnements externes peuvent venir :
- d'un problème dans la matière première. Par exemple, la rupture du stock d'une matière première, ou la non conformité de la qualité par rapport aux exigences de fabrication, ... ;
- des aléas de l'environnement du système. Par exemple, un court circuit causant une coupure de l'alimentation électrique d'un SAP, ... ;
- d'une modification de la commande par le client. Par exemple, un changement dans la spécification du produit, ...

Les dysfonctionnements internes peuvent être dus :
- à un problème physique dans la PC. Par exemple, une mauvaise communication, une panne du calculateur, ... ;
- à un problème logiciel dans la PC. Par exemple, un bogue de programmation, le plantage du système d'exploitation, ... ;
- à un problème dans la PO. Par exemple, la détérioration d'un composant physique, d'un capteur ou d'un actionneur, ...

Nos travaux de thèse se focalisent sur le diagnostic des dysfonctionnements affectant la partie opérative d'un SAP.

2.2.4 Les dysfonctionnements de la partie opérative

Les dysfonctionnements de la partie opérative sont dus à l'incapacité d'un ou plusieurs composants, dit *défaillants*, à remplir leur fonction. Les défaillances affectant les composants d'un SAP sont multiples et de différentes natures. Selon (Combacau et al., 2002; Deschamps, 2007), la défaillance d'un composant peut être à l'origine :

- d'une détérioration progressive due au vieillissement de ce composant. Par exemple, l'usure progressive d'une pièce mécanique ;
- d'une détérioration brusque, comme dans le cas d'une mauvaise manipulation d'un opérateur abîmant un composant ;
- d'une propagation de la défaillance d'un autre composant. Par exemple, suite à la défaillance d'un capteur, la PC peut générer de mauvaises actions causant la défaillance d'un autre composant.

Afin d'utiliser les SAP, tout en assurant la sécurité des hommes et des installations, il est indispensable de disposer d'un mécanisme de surveillance des défaillances. Ce mécanisme doit être capable d'identifier les éléments défaillants du système, permettant ainsi d'exécuter les actions de réparation adéquates, afin que le SAP retrouve ses performances nominales.

2.3 La problématique du diagnostic dans les SAP

Suite aux récentes révolutions technologiques dans le domaine industriel, on retrouve de plus en plus d'éléments complexes intégrés dans un SAP, notamment dans sa partie opérative. Cette complexité des SAP est accompagnée par des besoins croissants en termes de sûreté de fonctionnement et de sécurité. En effet, il est indispensable de mettre en oeuvre une fonction de surveillance permettant de réagir à temps, aux possibles dysfonctionnements. Une réaction immédiate aux dysfonctionnements permet d'une part, d'éviter de considérables pertes humaines et matérielles. D'autre part, elle permet d'échapper à la propagation des défaillances entraînant ces dysfonctionnements, facilitant ainsi le pistage des causes de ces défaillances.

Afin d'illustrer le principe d'une fonction surveillance, il est indispensable de présenter quelques terminologies.

2.3.1 Terminologie du diagnostic

La diversité des terminologies trouvées dans différents travaux fait que nous avons jugé important d'établir un lexique sur les termes qui seront utiles pour la compréhension du présent rapport. Nous présentons dans la suite quelques définitions extraites des références suivantes : (Villemeur, 1988; Combacau, 1991; Toguyeni, 1992; Lefebvre, 2000; Zemouri, 2003; Philippot, 2006; Deschamps, 2007)

Définition 1. Défaut : c'est une déviation du système par rapport à son comportement normal, qui ne l'empêche pas de remplir sa fonction. Un défaut est donc une anomalie qui concerne une ou plusieurs propriétés du système, pouvant aboutir à une défaillance et parfois même à une panne.

Définition 2. Dégradation : tout état qui se caractérise par une évolution irréversible des caractéristiques d'un système est une dégradation. La dégradation peut être liée à des facteurs directs, tels que l'usage, le temps..., ou à des facteurs indirects, tels que l'humidité, la température.... La dégradation peut aboutir à une défaillance, quand les performances du système sont en dessous d'un seuil d'arrêt défini par les spécifications fonctionnelles.

Définition 3. Défaillance : une défaillance est une anomalie altérant ou empêchant l'aptitude d'une unité fonctionnelle à accomplir la fonction souhaitée. Une défaillance correspond à un passage d'un état à un autre, par opposition à une panne qui est un état. Par abus de langage, cet état de panne on pourra l'appeler mode de défaillance.

Une défaillance implique l'existence d'un défaut, puisqu'elle aboutit à un écart entre la caractéristique mesurée et la caractéristique de référence. Inversement, un défaut ne conduit pas nécessairement à une défaillance. En effet, le système peut très bien conserver son aptitude à assurer une fonction requise, si les défauts qui l'affectent n'ont pas d'impacts significatifs sur la mission. Si une défaillance peut conduire à une cessation de l'exécution de la mission principale du système, ce dernier est déclaré en état de panne. Ainsi, la panne est toujours le résultat d'une défaillance.

Définition 4. Panne : c'est la conséquence d'une défaillance affectant le système, aboutissant à une interruption permanente de sa capacité à remplir une fonction requise et pouvant provoquer son arrêt complet. C'est la cause de l'apparition de symptômes. Deux types de pannes peuvent être distinguées :

- les pannes permanentes : une fois la panne est produite, elle nécessite une action de réparation.
- les pannes intermittentes : le système peut retrouver son fonctionnement nominal après l'occurrence de la panne. Une panne intermittente est généralement le résultat d'une dégradation partielle et progressive d'un composant du système, pouvant aboutir à une panne permanente.

Définition 5. Symptôme, Observation, Mesure :
Un symptôme correspond à une ou plusieurs observations qui révèlent d'un dysfonctionnement. Il s'agit d'un effet qui est la conséquence d'un comportement anormal.

Une observation est une information obtenue à partir du comportement ou du fonctionnement réel du système.

Une mesure est une observation élémentaire du fait qu'elle reflète une et une seule grandeur physique. Elle est représentée par une variable dont le contenu est l'image d'une grandeur physique. Son obtention s'effectue par l'intermédiaire de capteurs.

2.3.2 Surveillance et diagnostic

Les SAP sont généralement caractérisés par la complexité de leurs structures, puisqu'ils imbriquent de nombreux éléments complexes de la PO. Le taux d'apparition de pannes dans un système augmente en fonction de la complexité de sa structure, ce qui rend la tâche d'analyse de ces pannes difficile. Cette difficulté justifie la nécessité de disposer d'un système de surveillance permettant d'alerter l'opérateur en cas de pannes, afin de pouvoir décider à temps des actions correctives.

2.3.2.1 Surveillance

Un système de surveillance a comme première vocation d'émettre à partir des information générées par les capteurs, des alarmes (Valette et al., 1989) dont l'objectif est d'attirer l'attention de l'opérateur de supervision sur l'apparition d'un ou plusieurs événements susceptibles d'affecter le bon fonctionnement de l'installation, comme le dépassement d'un seuil de sécurité au niveau du remplissage d'un reservoir.

Dans le cadre de notre travail, nous considérons la surveillance comme un dispositif *passif*, dans le sens où ce dispositif n'influence pas le comportement du système à diagnostiquer. Un système de surveillance permet de **détecter** les défaillances en observant l'évolution du système, puis à les **diagnostiquer** en localisant les éléments défaillants et enfin identifier les causes premières. Nous pouvons distinguer deux fonctions principales dans la surveillance qui sont la *détection* et le *diagnostic*.

2.3.2.2 Détection

La fonction de détection permet de discerner tout écart du système par rapport à son état de fonctionnement normal. Autrement dit, elle permet de déterminer la présence de défauts dans un système. Pour assurer cette fonction, il est indispensable de pouvoir distinguer entre les situations normale et anormale. Cette fonction représente très souvent un sujet de débat concernant sa place. Dans certains travaux (Combacau et al., 2000; Boufaied, 2003), cette fonction est considérée comme un élément distinct de la fonction de diagnostic et plutôt une entité de la surveillance. D'autres travaux (Chow et Wilsky, 1984; Isermann, 1984) considèrent cette fonction comme une information primordiale et indissociable du diagnostic. Ainsi, ils définissent le diagnostic comme la détection, la localisation et l'identification de défauts.

2.3.2.3 Diagnostic

La fonction diagnostic permet de déterminer les causes et de localiser les éléments défaillants, qui ont entraîné la dégradation du système (Combacau et al., 2000). En effet, le diagnostic établit un lien de cause à effet entre un symptôme observé et la défaillance constatée. Cette fonction suit la fonction de détection et inclut les fonctions de localisation et d'identification.

Localisation de défauts : il s'agit de localiser le sous-système affecté par le défaut détecté, responsable de la défaillance du système. La localisation consiste, en effet, à remonter les symptômes pour retrouver l'ensemble des éléments défaillants. Ce problème est difficile à résoudre. En effet, il est possible de déterminer une défaillance, ou une panne, résultant d'un défaut. Par contre, le problème inverse est plus difficile à résoudre, puisque une panne peut résulter d'un ou plusieurs défauts, comme il est montré dans la figure 2.2, inspirée de (Zemouri, 2003).

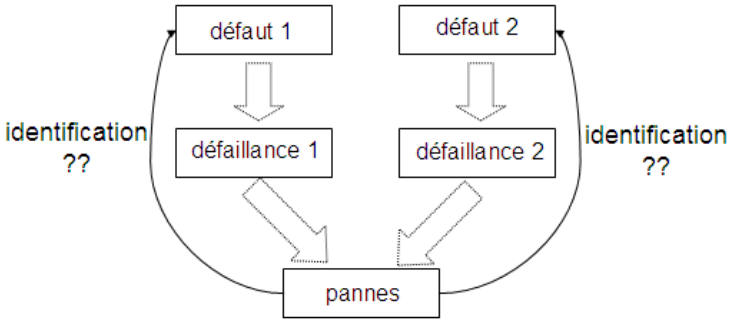

Fig. 2.2 – *La difficulté de localiser des défauts*

Identification de défauts : cette fonction suit la fonction localisation. Elle consiste à identifier les causes qui ont mené à une situation anormale.

Exemple 2.3.1
Afin de mieux clarifier les différentes notions de surveillance, détection, diagnostic, localisation et identification, nous considérons l'exemple d'une panne d'huile dans une voiture. Nous pouvons constater que la *dégradation* des performances de cette voiture apparaît suite à une surconsommation d'huile, tout en restant au dessous d'un seuil de consommation. A un certain moment, cette dégradation peut être accompagnée, quelquefois, par l'observation de *symptômes* de défaillances, comme le dégagement d'une fumée blanche. La *détection* correspond au dépassement d'un certain seuil de consommation, qui provoque le déclenchement d'une alarme indiquant l'occurrence d'une *défaillance*.

Le système de *diagnostic* permet de *localiser* le sous-système à l'origine de cette surconsommation (les segments du moteur, qui ont pour fonction d'assurer l'étanchéité du piston dans le cylindre) et d'*identifier* la cause de surconsommation (par exemple, l'usure des segments).

Mode de fonctionnement : un système présente généralement plusieurs modes de fonctionnement. On peut observer des modes de plusieurs types parmi lesquels (Zemouri, 2003) :

- **Mode de fonctionnement normal** (nominal) : dans ce mode, le fonctionnement du SAP est conforme aux exigences requises. Les performances du système coincident avec le cahier des charges établi par l'exploitant.
- **Mode de fonctionnement dégradé** : le SAP remplit partiellement les exigences requises. Malgré l'absence de défaillances, les performances du système sont inférieures à celles attendues par l'exploitant. Cette dégradation progressive des performances du SAP est généralement due au vieillissement d'un ou plusieurs composants du système.
- **Mode de défaillance** : le système passe à ce mode suite à l'occurrence d'une ou plusieurs défaillances. Les conséquences de ces défaillances sur le système caractérisent le mode de défaillance. En effet, un système peut avoir plusieurs modes de défaillances, par contre, il admet un seul mode de bon fonctionnement.

2.4 Classification des méthodes de diagnostic

Les méthodes de diagnostic des défauts utilisées dans le milieu industriel sont très variées. Leur principe général repose sur une comparaison entre les données observées au cours du fonctionnement du système et les connaissances acquises sur son comportement normal et ses comportements de défaillance (Combacau, 1991). Dans cette section, nous présentons une classification des principales méthodes de diagnostic rencontrées dans la littérature. Cette classification, représentée dans la figure 2.3, peut être réalisée selon plusieurs critères tels que la nature de l'information disponible (quantitative ou qualitative), la dynamique du système (continu, discret ou hybride), la structure de prise de décision (centralisée, décentralisée ou distribuée). Dans la suite, nous proposons une classification non exhaustive des méthodes de diagnostic selon deux axes : les *approches sans modèles* et les *approches à base de modèles* (Zwingelstein, 1995). Nous référons le lecteur aux travaux suivants pour avoir plus de détails : (Willsky, 1976; Isermann, 1984; Basseville, 1988; Combacau, 1991; Zemouri, 2003).

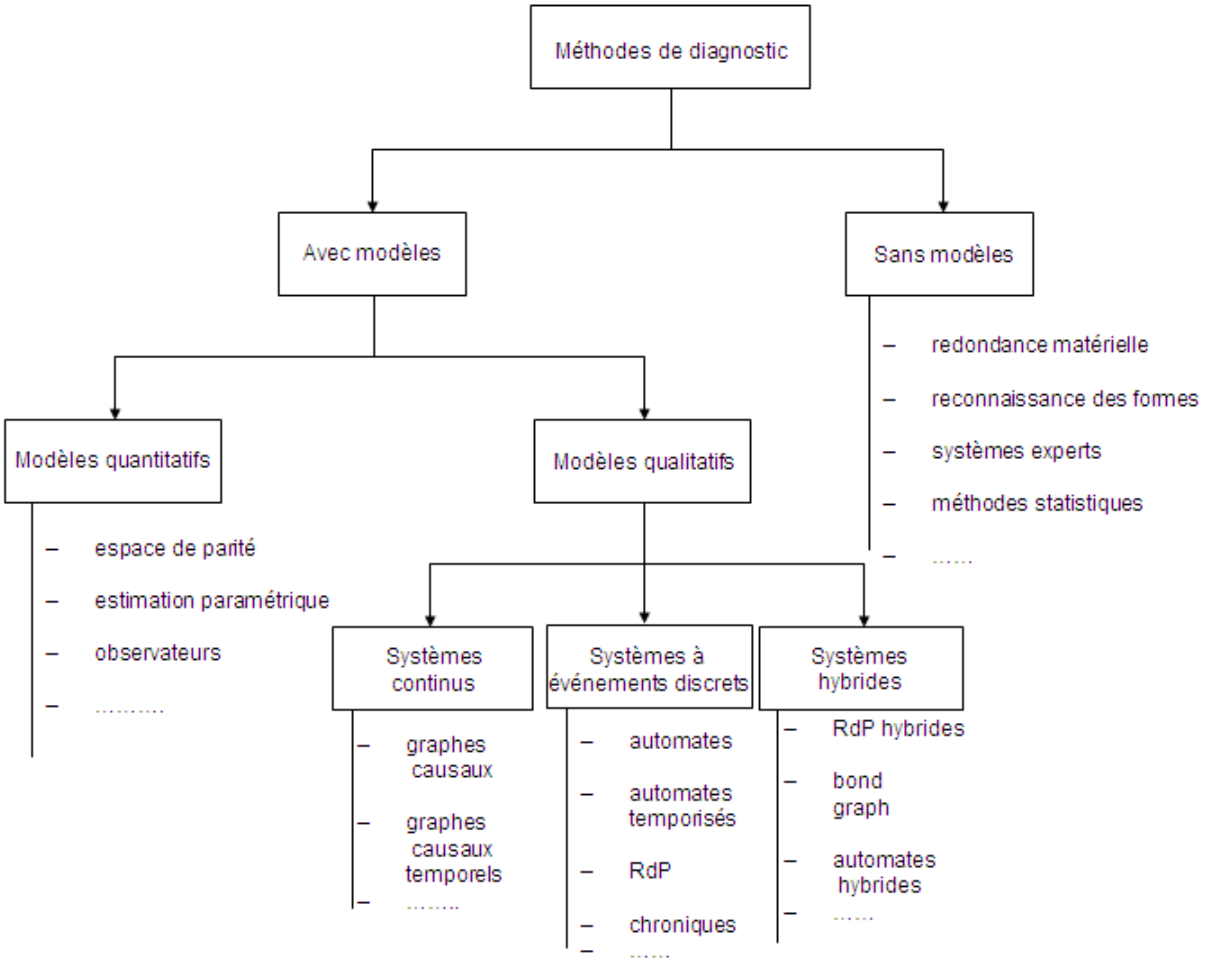

Fig. 2.3 – *Une classification des méthodes de diagnostic*

2.4.1 Méthodes sans modèles

Dans certaines applications industrielles, il est difficile, voire impossible, d'obtenir le modèle du système. Cette difficulté est justifiée par la complexité accrue ou à de nombreuses reconfigurations intervenants durant le processus de production. En effet, seules les méthodes de surveillance sans modèles sont opérationnelles pour ce type d'applications industrielles. Ces méthodes de diagnostic se basent sur des informations issues d'une expérience préalable, sur des règles heuristiques ou encore sur des exemples de résolution. Parmi ces méthodes, on trouve :

La méthode du seuillage : les signaux fournis par les capteurs sont comparées avec des valeurs limites constantes ou adaptatives (évoluant en fonction du point de fonctionnement) (Montmain, 1992). Un premier niveau indique la présence probable d'un défaut alors qu'un second niveau peut en caractériser la gravité. Le franchissement d'un seuil révèle la présence d'une anomalie.

Les méthodes statistiques : les méthodes statistiques supposent que les signaux fournis par les capteurs possèdent certaines propriétés statistiques, sur lesquelles des tests de seuil sont établis (Basseville, 1988; Zemouri, 2003). En effet, l'étude de l'évolution de la moyenne ou de la variance d'un signal peut favoriser la mise en évidence d'une anomalie.

La reconnaissance des formes : ces méthodes reposent sur l'utilisation des algorithmes de classification des formes et des mesures (continues ou discrètes). Le fonctionnement d'un système de diagnostic par reconnaissance des formes se déroule en trois phases (Dubuisson, 1990; Ondel, 2006) :

- une phase d'analyse qui consiste à déterminer et à réduire l'espace de représentation des données et à définir l'espace de décision permettant de spécifier l'ensemble des classes possibles ;
- une phase de choix d'une méthode de décision permettant de définir une règle de décision qui a pour fonction de classer les nouvelles observations dans les différentes classes de l'ensemble d'apprentissage ;
- une phase d'exploitation qui détermine, en appliquant la règle de décision, le mode de fonctionnement du système en fonction de chaque nouvelle observation recueillie sur le processus.

Les systèmes experts : les systèmes experts utilisent une information heuristique pour lier les symptômes aux défauts (Zwingelstein, 1995). Ce sont des systèmes à base de règles qui établissent des associations empiriques entre effets et causes (Farreny, 1989). Ces associations sont généralement fondées sur l'expérience de l'expert plutôt que sur une connaissance de la structure et/ou du comportement du système. Leur fonctionnalité est de trouver la cause de ce qui a été observé en parcourant les règles par un raisonnement inductif par chaînage avant ou arrière.

2.4.2 Méthodes à base de modèles

Ces méthodes reposent sur une comparaison du comportement du système avec le comportement du modèle qualitatif et/ou quantitatif établi (Combacau et al., 2000). Tout écart est alors synonyme d'une défaillance, comme indiqué dans le schéma de la figure 2.4.

Selon le type du modèle (qualitatif et/ou quantitatif), on peut distinguer deux branches de méthodes : les méthodes quantitatives issues de la communauté FDI (Failure Detection and Identification) et les méthodes qualitatives issues des communautés IA et SED. La dissociation entre les méthodes qualitatives et les méthodes quantitatives n'implique pas que ces deux aspects sont disjoints. En réalité, ces deux types d'approche peuvent coexister au sein d'une même méthode de diagnostic.

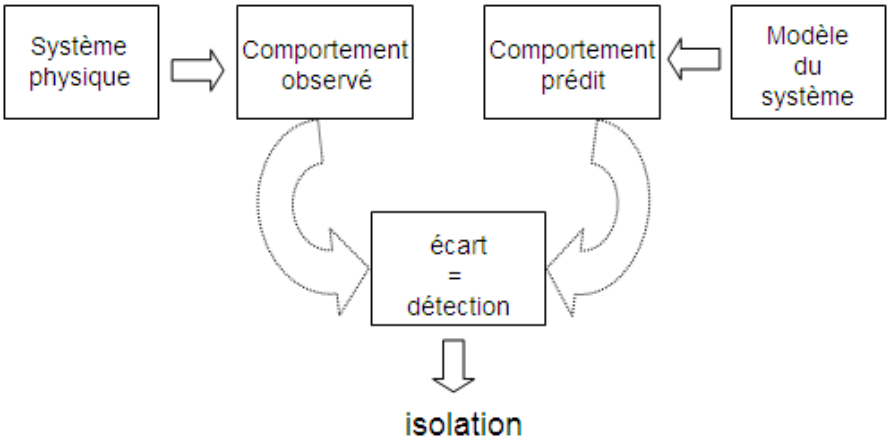

Fig. 2.4 – *Principe des méthodes de diagnostic avec modèles.*

2.4.2.1 Méthodes de diagnostic à base de modèles quantitatifs

Ces méthodes reposent sur l'estimation de l'état, des paramètres ou de l'espace de parité en utilisant des modèles mathématiques du système décrivant le comportement du système. Si l'écart entre ces modèles et les variables du système dépasse un certain seuil, une défaillance est alors détectée. A ce moment, un résidu sera généré et comparé avec toutes les signatures des défauts connues, afin d'isoler et d'identifier la défaillance. Parmi les différentes méthodes de détection et de diagnostic utilisant des modèles mathématiques, nous trouvons principalement l'espace de parité, les observateurs et l'estimation paramétrique.

La méthode d'espace de parité : La méthode de l'espace de parité a été une des premières méthodes employées à des fins de FDI (Chow et Wilsky, 1984; Gertler et Singer, 1990). Cette méthode repose sur la vérification de la cohérence (parité) des modèles du procédé avec les mesures issues de capteurs et des entrées connues (consignes, signal de commande, etc...).

La méthode à base d'observateurs :

Cette méthode se base sur la reconstruction de la sortie du processus à l'aide d'observateurs, de la comparer avec la sortie mesurée, puis à utiliser l'écart entre ces deux fonctions est utilisé comme résidu (Beard, 1971; Jones, 1973).

La méthode d'estimation paramétrique : cette méthodes suppose l'existence d'un modèle paramétrique décrivant le comportement du système et que les valeurs de ces paramètres en fonctionnement nominal soient connues. Elle consiste alors à identifier les paramètres caractérisant le fonctionnement réel, à partir de mesures des entrées et des sorties du système (Willsky, 1976). Pour détecter l'apparition de défaillances dans le

système, il faut effectuer la comparaison entre les paramètres estimés et les paramètres théoriques.

2.4.2.2 Méthodes de diagnostic à base de modèles qualitatifs

Les modèles qualitatifs permettent d'abstraire le comportement du procédé avec un certain degré d'abstraction à travers des modèles non plus mathématiques mais des modèles de type symbolique (Travé-Massuyès et al., 1997). Ces modèles décrivent d'une manière qualitative l'espace d'état continu du système. Contrairement aux modèles de type numérique, les modèles qualitatifs ne représentent pas la physique du système, mais ils le décrivent en terme de mode de fonctionnement. Les méthodes à base de modèles qualitatifs peuvent être classifiées selon le niveau d'abstraction considéré du système à diagnostiquer, à savoir, les systèmes continus, les systèmes à événements discrets ou les systèmes hybrides dynamiques. Pour les systèmes continus, les approches ont été développées à base de graphes causaux (Bousson et al., 1998) et de graphes causaux temporels (Mosterman, 2001). Le diagnostic des SED est basé sur l'utilisation de modèles discrets tels que les réseaux de Petri et les automates d'états finis, ... Dans ce contexte, Sampath et al. ont proposé dans (Sampath et al., 1995) une approche de diagnostic devenue une référence dans la littérature. Cette approche consiste à modéliser le système par des automates à états finis, puis à construire son diagnostiqueur. Pour les systèmes hybrides, on trouve des méthodes reposant sur des modèles hybrides tels que les automates hybrides à temps discret (Bhowal et al., 2007), les bond graphs (Tagina et al., 1995; Feenstra et al., 2001; Samantaray et Bouamama, 2008), les RdP hybrides (Gomaa, 1997)....

Dans le prochain chapitre, nous ferons un survol de ces approches. Nous insisterons sur les méthodes qui concernent directement le travail présenté dans ce mémoire.

2.4.2.3 Critères des méthodes de diagnostic à base de modèles

Les méthodes de diagnostic doivent tenir compte de certains critères qui varient en fonction des besoins en termes de sûreté, des ressources humaines et matérielles disponibles, de l'aspect critique du système surveillé, ... Dans (Philippot, 2006), plusieurs critères communs aux méthodes de diagnostic ont été dégagés. En effet, une méthode de diagnostic doit :

- fournir un diagnostic fiable (pas de fausses alarmes ni d'alarmes manquantes),
- être algorithmiquement concevable,
- être réalisable en temps réel,
- avoir un temps de réponse raisonnable (faible complexité),
- permettre un diagnostic rapide des défauts,

- être facile à mettre en oeuvre,
- nécessiter un nombre réduit de capteurs.

2.4.2.4 Approches de diagnostic centralisés, décentralisés et distribuées

La distribution des composants d'un système, des informations qu'il génère (commandes et compte-rendus des capteurs) peuvent parfois imposer le choix de la structure de prise de décision des méthodes de diagnostic. Ce choix peut être entre une structure centralisée, décentralisée ou distribuée.

La structure centralisée consiste à associer un modèle global du procédé avec un seul module de diagnostic. En conséquence, le module de diagnostic collecte les différentes informations du système avant de prendre sa décision finale sur son état de fonctionnement (Sampath et al., 1995). Cette structure se montre performante en terme de diagnostic. Cependant, elle est exposée au problème de l'explosion combinatoire des modèles utilisés surtout lorsqu'il s'agit de systèmes complexes. En effet, plusieurs approches de diagnostic, reposant sur des structures décentralisées et distribuées, ont été proposées dans la littérature.

La structure décentralisée se base sur un modèle global du système à qui sont associés plusieurs modules de diagnostic locaux. Chacun reçoit les informations observables qui lui sont spécifiques et prend une décision locale en se basant sur ses observations locales. Afin de lever le problème d'indécision et permettre aux modules de diagnostic locaux de diagnostiquer l'ensemble de défauts, un coordinateur (Debouk et al., 2000) doit être utilisé. Il traite les différentes décisions locales, communiquées par les modules locaux, afin de prendre une décision finale.

Dans la structure distribuée, le système est modélisé à travers ses composants par plusieurs modèles locaux. Chacun étant associé à un module de diagnostic local responsable de son composant. Un protocole de communication permet la communication directement entre les différents modules afin de gérer les conflits décisionnels (Silveira, 2003). Chaque module de diagnostic prend sa décision en se basant sur sa propre observation locale et celle communiquée par les autres modules (Xue et al., 2005).

Dans notre travail, nous nous intéressons uniquement aux structures centralisés. Des extensions aux autres structures peuvent être réalisées par la combinaison des démarches de décentralisation de distributions.

2.4.2.5 Diagnostic hors-ligne/en-ligne

Le diagnostic **hors-ligne** consiste à utiliser un ensemble de comportements et d'observations du système connus à l'avance, pour élaborer son diagnostic. Cette approche

n'impose pas des critère de réactivité de la réponse de diagnostic. En effet, aucune limitation sur le temps de réponse n'est exigée.

Le diagnostic **en-ligne** (Grastien, 2005) consiste à calculer le diagnostic du système pendant qu'il fonctionne, en s'appuyant sur les observations générées. Les approches de diagnostic en-ligne présentent des exigences en terme de réactivité de la réponse. Cela implique un temps de réponse rapide de la part du module de diagnostic. Le diagnostic en-ligne conduit généralement à une difficulté qui concerne la complexité du calcul. En effet, la réponse de diagnostic doit être livrée en temps-réel, le plus rapidement possible. Si le module de diagnostic ne parvient pas à gérer le flux d'observations et inférer le diagnostic suffisamment rapidement, alors le diagnostic en-ligne n'est plus possible. Vu que le nombre d'observations augmente régulièrement, il est nécessaire de disposer d'un diagnostic incrémental, où seul le diagnostic de la date t_{i-1} est pris en compte pour le calcul du diagnostic de la date t_i. En effet, la complexité de traitement ne doit pas dépendre du nombre d'observations reçus avant la date t_{i-1}, puisque ce nombre augmente d'une manière non bornée.

2.5 Conclusion

Nous avons présenté dans ce chapitre la problématique générale de surveillance des systèmes automatisés de production. A travers ce premier chapitre, nous avons souligné l'importance du rôle que joue un système de diagnostic dans le bon fonctionnement d'un processus industriel. La première partie de ce chapitre a été dédiée à la présentation de quelques terminologies, notions et mots clés utilisés dans ce mémoire de thèse. En effet, nous avons établi que le diagnostic d'un SAP se fait à travers trois fonctions de base : la détection, la localisation et l'identification. Dans une deuxième partie, nous avons présenté une classification des différentes méthodes de diagnostic. Deux grandes catégories de méthodes de diagnostic ont été abordées : méthodes à base de modèles et méthodes sans modèles. Nous avons constaté que les méthodes avec modèles reposent sur une comparaison du comportement observable du système avec son comportement prévu, décrit par un modèle qualitatif et/ou quantitatif. Les méthodes à base de modèles qualitatifs consistent à abstraire certains aspects continus du système, à travers une représentation symbolique, telles que les méthodes de diagnostic des SED et des SDH.

Dans le chapitre suivant, nous allons rappeler les principales approches de diagnostic à base de modèles autour desquels ce mémoire de thèse est focalisé. Nous focalisons notre étude sur les approches de diagnostic issues de la communauté SED. Nous étudions également quelques extensions de ces approches destinées aux SDH.

Chapitre 3

Diagnostic a base de modèles des SED et des SDH linéaires

3.1 Introduction

Le premier chapitre a mis en évidence l'importance du diagnostic des défaillances pour assurer le bon fonctionnement d'un système automatisé de production et éviter les pertes de biens et de vies humaines. Ce deuxième chapitre est consacré à la présentation d'un état de l'art sur les principales méthodes de diagnostic à base de modèles rencontrées dans la littérature, issues de la communauté automatique et la communauté IA. Cet état de l'art, permettant de positionner notre travail de recherche, ne se veut en aucun cas exhaustif.

Dans la première partie de ce chapitre, nous présentons les méthodes de diagnostic développées dans le cadre des Systèmes à Événements Discrets (SED). D'abord, nous faisons un rappel sur la classe des SED ainsi que leur principaux outils de modélisation. Ensuite, nous présentons les méthodes de diagnostic à base de modèles SED, les plus pertinentes pour la compréhension de notre travail. Nous considérons une classification des méthodes de diagnostic de SED basée sur la nature du modèle. En effet, nous pouvons distinguer deux catégories de méthodes : les méthodes basées sur des modèles logiques et celles basées sur des modèles temporisés.

Dans la deuxième partie, nous étudions quelques contributions développées dans le cadre du diagnostic des Systèmes Dynamiques Hybrides (SDH). Un rappel, non-exhaustif, de cette classe de systèmes ainsi que ses outils de modélisation et d'analyse sont présentés. Ensuite, nous survolons quelques méthodes de diagnostic reposant sur des modèles hybrides.

3.2 Cadre général du diagnostic des SED

Nous présentons dans cette section le cadre général du diagnostic des SED. Nous introduisons d'abord la classe des SED ainsi que les outils permettant sa modélisation. Ensuite, nous présentons le contexte général du diagnostic pour cette classe de systèmes.

3.2.1 Introduction aux SED

D'une manière informelle, un SED est un système dynamique à espace d'états discrets dont les transitions entre les états sont effectuées suite à l'occurrence d'événements. L'occurrence d'un événement est instantanée dans le sens où elle n'a pas de durée. Un SED demeure dans le même état en l'absence d'événements.

La classe des SED a été largement étudiée dans la littérature (Ramadge et Wonham, 1987; Lin et Wonham, 1988; Sampath et al., 1995). Cet intérêt est justifié par l'existence d'un grand nombre de systèmes réels évoluant d'une manière discrète.

Cassendras et Lafortune ont défini un SED dans (Cassandras et Lafortune, 1999) comme suit :

Définition 6. Un système à événements discrets est un système dont l'état évolue en fonction de l'occurrence d'événements asynchrones, sur une échelle de temps continue.

Afin de présenter l'intuition de cette définition, nous considérons l'exemple suivant :

Exemple 3.2.1

Nous considérons l'exemple d'un feu de circulation. Ce système peut être considéré comme un SED, si nous traitons que certains aspects de son fonctionnement à un niveau d'abstraction particulier. On suppose que ce dispositif comprend trois feux ayant les couleurs verte, rouge et orangée. A un instant donné, soit tous les feux sont éteints, soit un des trois feux est allumé. Ainsi, nous supposons que ce système admet les états suivants : **rouge**, **vert**, **orangé** ou **éteint**.

L'évolution de ce système est effectuée suite à l'occurrence de l'un des événements suivants : **circuler** (c), **stop** (s), **attention** (a) **éteindre** (e). La figure 3.1 présente une évolution possible de ce SED.

A son état initial (instant t_0), le système est supposé être à l'état **éteint**. L'occurrence de l'événement a à l'instant t_1 fait évoluer le système vers l'état **orangé**. De la même manière, le SED passe aux états **éteint**, **orangé**, **rouge** puis **vert**, suites aux occurrences respectives des événements e, a, s et c. En

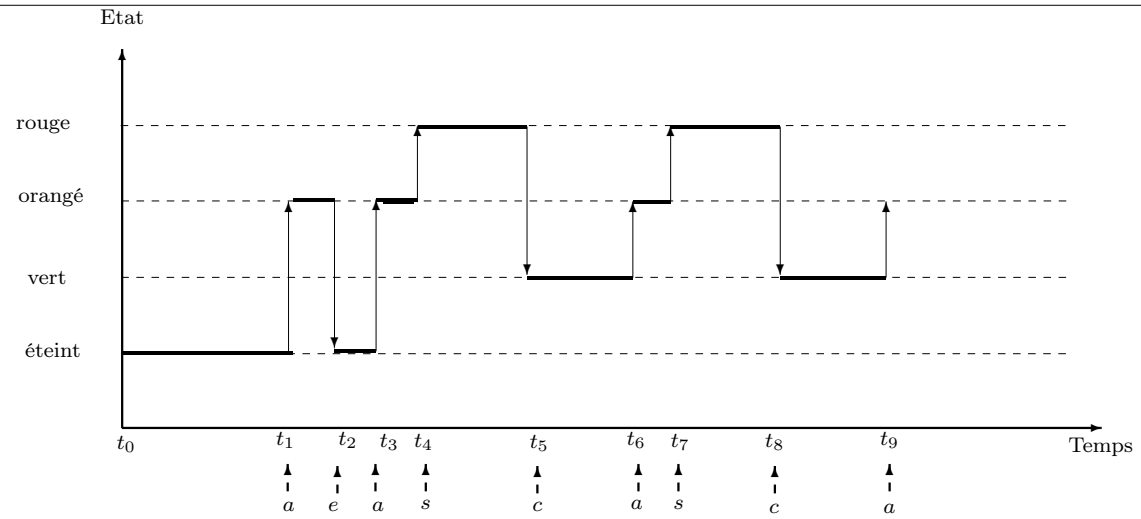

Fig. 3.1 – Chronogramme d'une évolution de SED dans le temps.

effet, l'évolution d'un SED peut être décrite par un ensemble de couples : (σ,t) où σ représente un événement (ou une action) et t représente l'instant de l'occurrence de cet événement. Dans notre exemple, une évolution possible du système peut être définie par la séquence suivante d'événements : (a,t_1), (e,t_2), (a,t_3), (s,t_4), (c,t_5),(a,t_6), (s,t_7)...

Cet ensemble ordonné de couples constitue ce que l'on appelle une *trace* (mot ou *trajectoire*) du système. Dans une telle description de l'évolution du système, l'information temporelle est représentée d'une manière explicite, ainsi, cette trace est dite *temporisée*. Il est possible de décrire l'évolution du système en faisant abstraction du temps, qui sera représenté uniquement par l'ordre d'occurrence des événements. Une telle description du SED est dite *logique*. Dans l'exemple précédent, la trajectoire $a\ e\ a\ s\ c\ a\ s$, décrit une évolution logique du SED.

3.2.2 Outils de modélisation des SED

Nous présentons dans la suite quelques outils de modélisation des SED. Durant cette présentation, nous distinguons deux catégories de modèles pour les SED : les **modèles logiques** et les **modèles temporisés**.

3.2.2.1 Les modèles logiques

Ces modèles permettent de représenter l'ordre logique d'occurrence des événements dans un SED. Ainsi, le temps est décrit implicitement dans une trajectoire et seul l'ordre d'occurrence des événements constituant une trajectoire est pris en considération. Les

modèles logiques sont utilisés pour l'étude des propriétés qualitatives des SED. Parmi ces modèles, nous trouvons les automates à états finis et les réseaux de Petri.

3.2.2.1.1 Les automates à états finis Un *automate à états finis* est une machine à états qui permet de décrire les évolutions possibles d'un système à événements discrets (Cassandras et Lafortune, 1999). Ainsi, le comportement d'un SED est représenté à travers un ensemble d'événements associé à un ensemble d'états. Formellement, un automate à états finis est défini par un quintuplet

$$G = (Q, \Sigma, \delta, q_0, Q_m)$$

où :

- Q est un ensemble fini d'états ;
- Σ est un ensemble fini d'événements (ou de symboles) ;
- δ est une fonction de transition, $\delta : Q \times \Sigma \to Q$;
- $q_0 \in Q$ est un état initial ;
- $Q_m \subseteq Q$ est l'ensemble d'états finaux.

Un automate à états finis est dit *déterministe* si à partir d'un état donné, au plus, une seule transition est possible sur l'occurrence d'un événement.

Exemple 3.2.2
Dans la figure 3.2.1, nous modélisons le comportement du SED introduit dans l'exemple 3.2. Chaque sommet de l'automate correspond à un état du SED. Les transitions entre ces états sont représentées par des arcs. Chaque arc, reliant deux sommets, est étiqueté par un événement de transition entre les états correspondants à ces sommets. L'état initial du système **éteint** est marqué par une flèche.

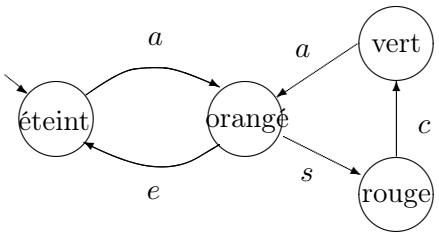

Fig. 3.2 – Exemple d'un automate à états finis.

La théorie des automates à états finis a été conjointement développée avec la théorie des langages. Nous présentons dans la suite quelques notions relatives à la théorie des langages, qui seront utilisées plus loin dans ce mémoire.

Un *mot* défini sur un alphabet Σ est une suite finie d'éléments de Σ. L'opérateur $||s||$ indique le nombre d'éléments de Σ dans le mot s.

Nous désignons par Σ^* l'ensemble de mots de longueur quelconque que l'on peut construire sur Σ. Un *langage* défini sur un alphabet Σ correspond à un sous-ensemble de Σ^*.

Soit s un mot défini sur un alphabet Σ. Un mot s_1 est dit *préfixe* de s s'il existe un mot $s_2 \in \Sigma^*$ telle que $s = s_1 s_2$. La préfixe-clôture $\overline{L}$ d'un langage L est le langage constitué par tous les préfixes des mots de L. Un language L est dit *préfixe-clos* s'il est égal à sa préfixe-clôture ; i.e., $L = \overline{L}$.

Un mot est dit *accepté* par un automate si, partant de son état initial et recevant successivement les symboles du mot d'entrée, l'automate évolue vers un état final. Un langage $L(G)$ accepté par un automate G est constitué par l'ensemble des mots acceptés par l'automate ; i.e., $L(G) = \{\sigma \in \Sigma^* \mid \delta(q_0, \sigma) \in Q_m\}$, où δ désigne la fonction de transition sur l'occurrence d'un mot. Un automate G est dit *générateur* si tous ses états sont finaux ; i.e., $Q = Q_m$. Il est clair que le langage $L(G)$ accepté par un générateur $G = (Q, \Sigma, \delta, q_0)$ est préfixe-clos.

3.2.2.1.2 Les Réseaux de Petri Le modèle Réseau de Petri (RdP) a été introduit en 1964 par C. A. Petri (Petri, 1962). Il constitue un outil de modélisation de SED particulièrement adapté pour spécifier le comportement des systèmes industriels. Il permet de modéliser et de visualiser des primitives de comportement telles que la synchronisation, le parallélisme, le partage de ressources ou le séquencement (David et Alla, 1989). Les RdPs sont représentés autour d'un langage graphique et d'un langage mathématique. Un RdP se présente sous la forme de places et de transitions, reliées par des arcs.

Cet outil possède de nombreuses extensions comme les RdPs colorés (David et Alla, 1989), ou les RdP temporels (Merlin, 1974; Berthomieu et Diaz, 1991). Cette richesse d'extensions fait que les RdPs peuvent s'appliquer à la plupart des phases de développement d'un système, de la spécification de la commande à la supervision en passant par le diagnostic et la validation (Philippot, 2006).

3.2.2.2 Modèles temporisés

Dans certaines applications, l'information temporelle est indispensable et doit être considérée explicitement dans le modèle du SED. Les modèles ayant cette propriété sont dits temporisés. Parmi ces modèles, nous pouvons citer les automates temporisés et les RdPs temporels, qui seront décrits brièvement dans ce qui suit. Nous notons qu'une description plus détaillée de l'outil automate temporisé sera présentée dans le chapitre suivant.

3.2.2.2.1 Les automates temporisés Les automates temporisés ont été introduits par R. Alur et D. Dill dans les années 1990 (Alur et Dill, 1994). Il s'agit d'automates classiques munis d'un ensemble de variables réelles, appelées *horloges*, qui évoluent de manière continue et synchrone avec le temps. A chaque transition est associée une condition sur la valeur des horloges, dite *garde*, décrivant quand une transition peut être exécutée, et un ensemble d'horloges remises à zéro lors du franchissement de la transition. Chaque état discret contient un invariant (une contrainte sur les horloges) qui restreint le temps d'attente dans l'état et donc force l'exécution d'une transition. Dans ce qui suit, nous présentons l'exemple d'un automate temporisé.

L'automate temporisé présenté dans la figure 3.3 décrit le comportement temporel du SED considéré dans l'exemple 3.2.1. Ce modèle augmente le modèle automate à états finis dans la figure 3.2, par une horloge, notée x, permettant de mesurer le temps écoulé dans chaque état discret du système. Cette horloge est mise à zéro après chaque transition entre deux sommets. La contrainte d'invariance restreint le temps de séjour possible dans chaque état discret du système. En effet, la contrainte $x \leq 20$ implique que le système ne peut pas séjourner dans l'état discret **vert** plus que 20 unité de temps (u.t.). Lorsque l'horloge x atteint 20 u.t., une transition vers l'état **orangé** sera franchie, sur l'occurrence de l'événement a. La garde de cette transition ($x = 20$) est évidement satisfaite par la valeur de x.

L'état d'un automate temporisé est l'association d'un sommet de l'automate (état discret) avec l'ensemble de valeurs réelles des horloges à un instant donné (état continu). La transition du sommet **vert** vers le sommet **orangé** peut être franchie uniquement à partir de l'état $(vert, \langle x = 20 \rangle)$.

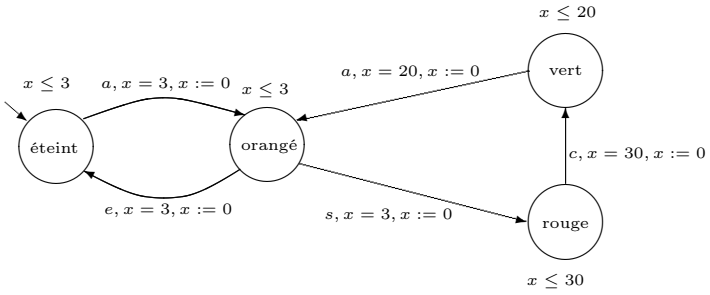

Fig. 3.3 – Exemple d'un automate temporisé.

Le modèle automate temporisé représente un formalisme relativement simple à manipuler. Toutefois, il possède l'expressivité nécessaire pour la modélisation de systèmes temporisés. En plus, Ce formalisme offre une grande capacité d'analyse comportementale des systèmes à travers les différentes méthodes d'analyse développées dans la littérature, telles que le model-checking (Yovine, 1998) ou l'analyse d'accessibilité (Alur, 1999; Bengtsson et Yi., 2004).

3.2.2.2.2 Les réseaux de Petri temporels Le modèle réseau de Petri temporel (Merlin, 1974) est une extension du modèle réseau de Petri qui associe deux dates min et max à chaque transition. En effet, si une transition t a été sensibilisée pour la dernière fois à une date θ, alors t ne peut pas être franchie avant la date $\theta + min$ ou après la date $\theta + max$, sauf si une autre transition a désensibilisé t avant que celle-ci ne soit franchie. Les RdPs temporels expriment des spécifications "en délais" qui sont largement utilisées dans la description de l'évolution temporelle des processus industriels. Plusieurs travaux se sont intéressés à la traduction du modèle RdP temporel vers le modèle automate temporisé (Sava et Alla, 2001; Cassez et Roux, 2006) afin d'exploiter la grande capacité d'analyse que représente le modèle automate temporisé.

3.2.3 Modélisation des défauts

Comme nous avons indiqué au cours du chapitre précédent, un diagnostic à base de modèles consiste à comparer le comportement prévu du système décrit par un modèle et celui réellement observé. Toute discordance entre les deux indique la présence d'au moins un défaut, que l'on cherchera à localiser et identifier. En effet, afin d'appliquer une telle méthodologie de diagnostic, il faut disposer d'un modèle qui décrit le comportement nominal et de défaut du système. Ce modèle de défaut doit décrire les comportements défaillants possibles du système.

La modélisation des défauts suppose l'acquisition d'une connaissance **a priori** des défauts que l'on souhaite diagnostiquer. Dans la définition 3, nous avons souligné qu'une panne du système correspond à un état de dysfonctionnement, tandis qu'une défaillance, ou un défaut source d'une défaillance, correspond à un événement, qui peut mener vers un état de panne. Dans le contexte des SED, l'apparition d'une panne correspond au passage vers un état de panne. Ce passage peut être modélisé par une transition sur un défaut, où on considère une **modélisation à base d'événements** (Lin et Wonham, 1994; Sampath et al., 1995). Si l'on considère une **modélisation à base d'états** (Zad et al., 1998, 1999), une partition des états du système en états nominaux et états de panne est préalablement établie. Dans ce dernier cas, un système est déclaré en panne s'il atteint un état de panne.

Dans le cadre des pannes permanentes, le passage vers un état de début de panne se fait suite à l'occurrence d'un événement de défaut. Le système va évoluer ensuite vers d'autres états de pannes. S'il s'agit de pannes intermittentes, le système peut retourner vers un état de fonctionnement normal, suite à l'occurrence d'un événement de retour en fonctionnement normal.

3.2.4 Notion d'observabilité

La notion d'observabilité dans le cadre des SED a été introduite dans les travaux de Lin et Wonham (Lin et Wonham, 1988). En effet, les événements qui décrivent l'évolution d'un SED ont été classés en événements **observables** et événements **non observervables**.

L'ensemble des événements observables, noté Σ_o, correspond aux événements issus des capteurs physiques et des actionneurs d'un procédé. Ces événements peuvent être observés par l'environnement du système, notamment par le module de surveillance.

L'ensemble des événements non observables, noté Σ_{uo}, correspond aux événements internes du système (des informations échangées entre les composants élémentaires de la PO) et aux événements de défauts dont les occurrences ne peuvent pas être directement observées. Aucune information directe sur l'occurrence d'un événement observable ne peut remonter à la PC, ou à l'environnement du système (Sampath et al., 1995). Néanmoins, il est possible d'inférer, indirectement, l'occurrence de ces événements en s'appuyant sur le comportement observable qui suit l'occurrence de tels événements.

Dans le cadre de notre étude, nous traitons les défauts correspondant à des événements non observables. Ainsi, nous écartons la surveillance des défauts qui peuvent être directement observés par des capteurs de défaut. En effet, nous considérons que le problème de surveillance devient dans ce cas trivial, puisqu'il suffit de transmettre le signal fourni par le capteur de défaut au module de surveillance pour qu'il déclenche une alarme. Cependant, il faut noter que le déploiement de tels capteurs n'est pas toujours possible. En effet, certains défauts n'admettent pas de capteurs physiques permettant de détecter leurs occurrences. Cela peut être dû à des limitations de nature technologique ou conceptuelle. En plus, ces capteurs ont souvent un coût très élevé et impliquent parfois des efforts supplémentaires en terme de maintenance.

3.3 Les méthodes de diagnostic des SED

La notion de diagnostic des SED a été introduite au milieu des années 1990 dans les travaux de sampath (Sampath et al., 1995, 1996). Plusieurs extensions de ces travaux ont été proposées dans le cadre des modèles SED temporisés (Tripakis, 2002; Zad et al., 2005; Lunze, 2006), des modèles hybrides (Fourlas et al., 2002; Bhowal et al., 2007), des architectures décentralisées (Debouk et al., 2000) et distribuées (Silveira, 2003; Xue et al., 2005), des modèles stochastiques (Thorsley et Teneketzis, 2005), ...

Dans la suite, nous présentons les méthodes de diagnostic de SED les plus pertinentes pour la compréhension de notre travail. Nous considérons une classification des méthodes de diagnostic de SED basée sur la nature du modèle. En effet, nous pouvons distinguer

deux catégories de méthodes :

- les méthodes basées sur des modèles logiques ;
- les méthodes basées sur des modèles temporisés.

Les méthodes à base de modèles logiques consistent à élaborer le diagnostic du système à partir d'un modèle logique du SED. Ainsi, le temps sera considéré uniquement d'une manière qualitative à travers l'ordre d'occurrence des événements. Le diagnostic résultant considérera le temps, de même, d'un point de vue qualitatif.

Les méthodes à base de modèles temporisés consistent à élaborer le diagnostic du système à partir d'un modèle temporisé du SED. Ainsi, l'aspect temporel sera considéré d'une manière explicite et quantitative, à travers l'utilisation d'horloges internes (Alur et Dill, 1994) ou la discrétisation du temps (Zad et al., 2005).

3.3.1 Méthodes de diagnostic à base de modèles logiques des SED

Nous présentons dans la suite les approches de diagnostic des SED reposant sur un modèle logique du système à diagnostiquer. Dans ce cadre, nous pouvons distinguer deux principales approches. La première approche, issue des travaux de Sampath (Sampath et al., 1995), se base sur une représentation événementielle des défauts. La seconde approche a été introduite par Zad (Zad et al., 2003) et repose sur modélisation à base d'état des défauts. Nous présentons dans la suite un aperçu de ces deux approches, tout en insistant sur l'approche de référence développée dans les travaux de Sampath.

3.3.1.1 Approche issue de la contribution de Sampath

Dans la littérature, les travaux de Sampath et al. (Sampath et al., 1995, 1996) sont devenus une référence dans le domaine du diagnostic à base de modèles des SED. Dans (Sampath et al., 1995), une approche de modélisation et de diagnostic des systèmes complexes, basée sur une représentation logique des SED, a été proposée. Cette approche consiste à inférer les occurrences des événements des défauts non observables en utilisant les événements observables générés par le système.

Dans la figure 3.4, nous illustrons le principe de cette approche.

1. Dans une première étape, un modèle "complet", qui décrit le comportement normal et anormal du système, est construit. Ce modèle correspond à un automate à états finis, construit à partir d'une composition des modèles élémentaires des composants. Les défauts sont représentés par des événements non observables. A partir de ce modèle global, un outil de diagnostic, appelé *diagnostiqueur*, est compilé hors-ligne, sous la forme d'un automate à états finis déterministes.

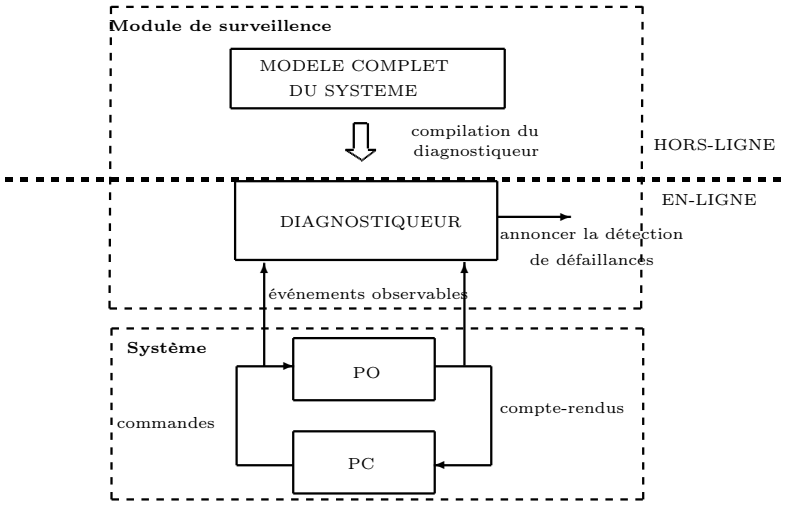

Fig. 3.4 – *Principe de l'approche de Sampath*

2. Dans la deuxième étape, le diagnostiqueur est déployé en-ligne ; i.e., pendant le fonctionnement du système. L'automate du diagnostiqueur intercepte, en entrée, les événements observables générés par le système, et fournit une estimation de son état courant et des défauts affectant son fonctionnement.

La construction du modèle complet du système commence par la modélisation de chaque composant du système par un automate à états finis. Ainsi, le générateur $G_i = (Q_i, \Sigma_i, \delta_i, q_{0_i})$ décrit le comportement du $i^{ème}$ composant du système. Les états X_i et les événements Σ_i reflètent le comportement normal et anormal du $i^{ème}$ composant. Pour obtenir le modèle global du système, il suffit alors de réaliser une composition des modèles des composants avec celui de la commande. L'automate résultant est noté $G = (Q, \Sigma, \delta, q_0)$. L'ensemble des événements $\Sigma = \Sigma_o \cup \Sigma_{uo}$ est réparti en deux sous-ensembles : le sous-ensemble Σ_o des événements observables et le sous-ensemble Σ_{uo} des événements non observables. L'automate générateur G doit vérifier deux conditions :

C_1) le langage L, accepté par G, est vivant ; i.e., il existe une transition de sortie à partir de chaque état $q \in Q$;

C_2) l'automate G ne contient pas de cycles formés exclusivement d'événements non observables.

On note par $\Sigma_f \subseteq \Sigma_{uo}$ l'ensemble des événements des défauts à diagnostiquer. On suppose que l'ensemble des défauts Σ_f forme une partition de m sous-ensembles de défauts $\Sigma_f = \Sigma_{f_1} \cup \cdots \cup \Sigma_{f_m}$, notée $\Pi_f = \{\Sigma_{f_1}, \ldots, \Sigma_{f_m}\}$, $m \geq 1$. Chaque sous-ensemble $\Sigma_{f_{j \in \{1,\ldots,m\}}}$ correspond à un groupe de défauts ayant le même effet sur le système, ou affectant le même composant de la PO.

A partir du modèle $G = (Q, \Sigma, \delta, q_0)$, un algorithme de synthèse du diagnostiqueur est appliqué, en considérant la partition de défauts Π_f, pour obtenir l'automate du diagnostiqueur $G_d = (Q_d, \Sigma_o, \delta_d, q_{d_0})$. Chaque état du diagnostiqueur est composé d'un

ensemble de paires de la forme (état,étiquettes). Chaque paire contient un état estimé du système, associé à un ensemble d'étiquettes éléments de $\{F_1, \ldots, F_m, N\}$. L'étiquette N indique que le système admet un fonctionnement normal s'il se trouve dans l'état estimé, associé à cette étiquette. Si un état estimé est associé à une étiquette F_i alors un défaut de l'ensemble Σ_{f_i} s'est produit, avant d'atteindre cet état. Si l'étiquette F_i est associée à tous les éléments d'un état du diagnostiqueur, alors cet état du diagnostiqueur est dit F_i-*certain*. Un état du diagnostiqueur est dit *normal*, si les étiquettes appartenant à cet état ne comportent que l'étiquette N et dit F_i-*incertain* s'il ne coïncide à aucun des deux cas précédents.

Lorsque le diagnostiqueur évolue vers un état F_i-certain, suite à l'observation d'une séquence d'événements, il annonce par le biais d'une fonction de décision, l'occurrence d'un défaut de l'ensemble Σ_{f_i}. Intuitivement, toute trajectoire du système, dont la projection observable est identique à la séquence d'événements observables générée par le système, contient au moins un défaut de l'ensemble Σ_{f_i}. Autrement dit, toutes les trajectoires estimées du système, qui expliquent le comportement observé, contiennent un défaut de Σ_{f_i}. Si le diagnostiqueur évolue vers un état F_i-incertain, aucune décision certaine sur l'occurrence d'un défaut ne pourra être prise. Ceci peut être expliqué par l'existence d'au moins deux trajectoires qui expliquent le comportement observé, l'une contient un défaut de l'ensemble Σ_{f_i} et l'autre non.

Afin de décrire le fonctionnement du diagnostiqueur, nous considérons l'exemple suivant.

Exemple 3.3.1

La figure 3.5 illustre l'exemple d'un modèle et son correspondant diagnostiqueur. Dans le modèle considéré, les transitions sur les événements **non observables** sont représentées par des **arcs en pointillés**. On suppose que $\Sigma_o = \{a, b, c, d, e\}$ et $\Sigma_{uo} = \Sigma_f = \{f_1, f_2\}$. On définit la partition de défauts de la manière suivante : $\Pi_f = \{\Sigma_{f_1}, \Sigma_{f_2}\}$, avec $\Sigma_{f_1} = \{f_1\}$ et $\Sigma_{f_2} = \{f_2\}$.

L'état $1N$ correspond à l'état initial du diagnostiqueur. Ainsi, on suppose que le système ne contient pas de défaut à son état initial. A partir de cet état et suite à l'observation de l'événement a, le diagnostiqueur évolue vers l'état F_1-incertain $\{3F_1\ 7N\}$. Cet état indique que le modèle du système est soit dans l'état 7 et aucun défaut ne s'est produit, soit dans l'état 3 avec l'occurrence d'un défaut de l'ensemble Σ_{f_1}. Ensuite, le diagnostiqueur évolue vers l'état $\{4F_1\ 9F_2\ 12N\}$ suite à l'observation de l'événement b.

Après chaque occurrence d'un événement observable, le diagnostiqueur estime les états atteignables du modèle, sur l'occurrence de toute possible séquence d'événements non observables suivie de l'événement observé. Cette estimation est accompagnée d'une propagation des étiquettes de défauts.

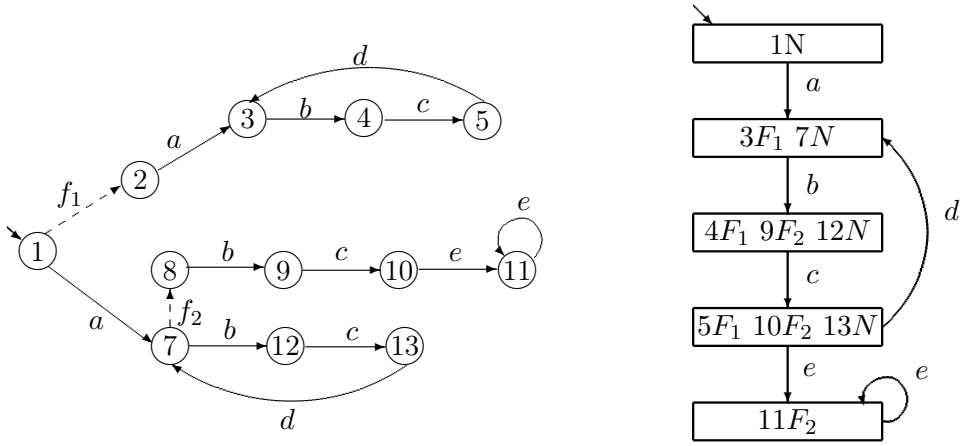

FIG. 3.5 – *Exemple d'un modèle automate à états finis G et son diagnostiqueur G_d*

Remarque 1. Nous pouvons dégager une similarité entre la construction du diagnostiqueur et la procédure de déterminisation d'un automate à états finis muni de ϵ-transitions (Cassandras et Lafortune, 1999). En effet, chaque événement non observable peut être considéré comme un événement ϵ.

En observant la séquence d'événements *abce*, à partir de l'état initial, le diagnostiqueur évolue vers les états $\{3F_1\ 7N\}$, $\{4F_1\ 9F_2\ 12N\}$, $\{5F_1\ 10F_2\ 13N\}$ et enfin $\{11F_2\}$. Il est clair que l'état $\{11F_2\}$ est F_2-certain. En effet, la fonction de décision annonce l'occurrence d'un défaut de l'ensemble Σ_{f_2}.

Si nous considérons, dans un deuxième exemple, l'observation de la séquence d'événements $abcdbcd(bcd)\ldots$. Dans ce cas, le diagnostiqueur évoluera, en permanence, dans un cycle d'états F_1-incertain. En effet, si un défaut se produit, le diagnostiquer ne peut pas inférer son occurrence quelque soit la longueur de la séquence d'événements observables qui suit ce défaut. On dit dans ce cas que le diagnostiqueur ne répond pas au critère de *diagnosticabilité*.

Pour que le diagnostiqueur soit capable d'isoler chaque défaut appartenant à un ensemble donné de défauts, le modèle du système doit répondre au critère de diagnosticabilité. Intuitivement, il faut pouvoir vérifier si ce dernier dispose d'informations en quantité suffisante pour pouvoir effectuer le diagnostic du système. Dans ce cas, le diagnostiqueur est capable d'identifier tout mode de défaillance, déclenché par l'occurrence d'un défaut, au bout d'un nombre fini d'événements suivant cette occurrence. Nous présentons, plus loin dans ce chapitre, une étude complète sur la diagnosticabilité des SED, ainsi que les méthodes développées pour la vérification de ce critère.

3.3.1.2 Approche issue de la contribution de Zad

Une méthode de diagnostic de SED, inspirée de l'approche de Sampath, a été proposée dans les travaux de Zad (Zad et al., 1998, 2003). Cette approche repose sur une modélisation à base d'états des défauts. Ainsi, l'identification de défauts repose sur l'atteignabilité d'états défaillants.

Le diagnostiqueur est composé d'un ensemble d'états enrichis indiquant les sorties observées, les états correspondants et les étiquettes décrivant les modes de fonctionnement possibles du système. Les sorties observées correspondent aux événements générés par des capteurs. Ces capteurs sont supposés être robustes et non susceptibles d'être défaillants. Nous notons que, contrairement à l'approche de Sampath, cette approche ne nécessite pas l'initialisation du diagnostiqueur en même temps que le système.

Nous illustrons dans la figure 3.6, un exemple inspiré de (Zad et al., 1999), qui présente un modèle SED et son diagnostiqueur.

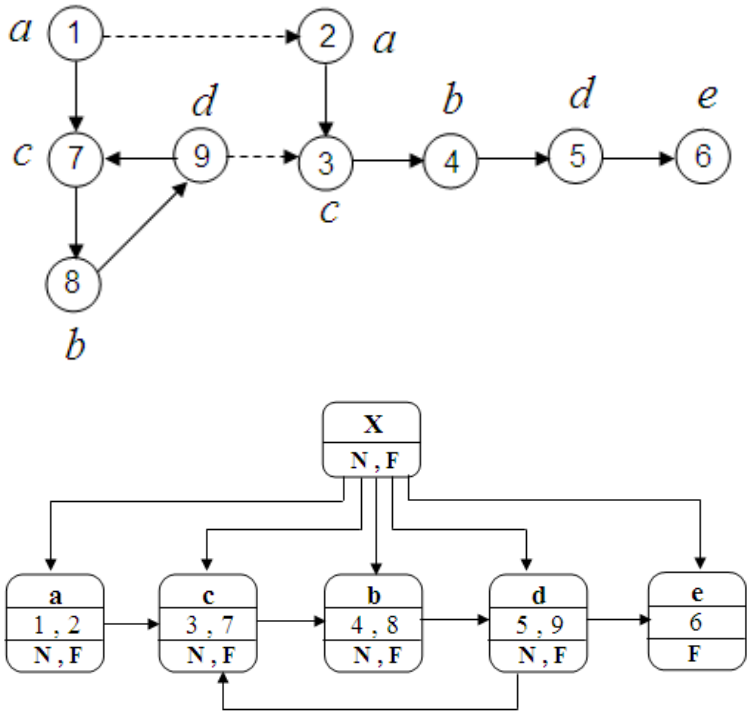

Fig. 3.6 – Diagnostiqueur construit selon l'approche de Zad.

Chaque état du modèle du système ainsi que de son diagnostiqueur est étiqueté par la sortie observée. L'état initial du diagnostiqueur correspond au sommet $(X, \{N, F\})$ qui signifie que l'état courant du système est inconnu et qu'il peut être normal ou associé à une défaillance. Supposons que la sortie d est observée, le diagnostiqueur évolue vers le sommet $(\{5, 9\}, \{N, F\})$ qui correspond à un état incertain (normal ou défaillant). Si

par la suite, la sortie e est observée, le diagnostiqueur évolue vers le sommet $(\{6\}, \{F\})$, un défaut est alors détecté.

3.3.2 Approches de diagnostic à base de modèles temporisés

Ces approches reposent sur des modèles représentant, d'une manière explicite, les relations temporelles liant les occurrences des événements dans un SED. De nombreuses méthodes se sont intéressées au diagnostic basé sur des modèles temporisés de SED. Ces méthodes visent à exploiter au maximum les contraintes temporelles qui existent entre les événements du SED, afin de permettre la discrimination des défauts.

La notion de signature temporelle est une notion intéressante pour la caractérisation temporelle des trajectoires discrètes de comportement normal et de défaut au sein d'une description événementielle (Philippot, 2006). En effet, il est possible que l'occurrence d'un défaut n'affecte pas l'ordre des événements observables qui suivent le défaut, par contre, elle affecte les dates d'occurrence de ces événements. Ainsi, l'exploitation du temps devient nécessaire dans ce cas pour caractériser les trajectoires des comportements défaillants.

L'intégration des contraintes temporelles dans le diagnostic des SED a été abordée par plusieurs chercheurs utilisant des formalismes mathématiques différents, tels que les automates temporisés (Alur et Dill, 1994), les RdP temporels (Merlin, 1974), les chroniques (Dousson, 1994), ... Nous proposons dans la suite, quelques méthodes de diagnostic de SED à base de modèles temporisés. Nous dégageons dans ce cadre deux types de méthodes qui se distinguent par l'élaboration ou non d'un pré-calcul de diagnostic : les méthodes **en-ligne** ou les méthodes **hors-ligne**.

3.3.2.1 Approches hors-ligne

Les approches hors-ligne sont caractérisées par l'élaboration d'un pré-calcul hors-ligne, afin de minimiser le calcul à effectuer pendant la phase en-ligne (à ne pas confondre avec le diagnostic hors-ligne qui est élaboré sur la base d'observations connues à l'avance). L'approche de Sampath fait partie de cette sous-classe d'approches. En effet, le calcul effectué hors-ligne consiste à estimer, pour chaque observation pouvant être générée par le système, l'ensemble des états du système et des défauts produits. L'ensemble de ces estimations est compilé sous la forme d'un automate à états finis, qui correspond au diagnostiqueur. Ainsi, une fois exécuté, le diagnostiqueur évolue d'une manière déterministe sur l'occurrence de chaque événement observable généré par le système, et fournit l'ensemble des défauts pouvant affecter son fonctionnement. En effet, le calcul effectué pendant la phase en-ligne se réduit au franchissement déterministe d'une transition associée à l'événement observé.

Nous présentons dans la suite quelques approches de diagnostic hors-ligne, basées sur des modèles temporisés de SED.

3.3.2.1.1 Approche de diagnostic à base de modèles à temps discret Zad a proposé dans (Zad et al., 2005), une extension temporisée de l'approche de diagnostic élaborée dans (Zad et al., 2003). Cette extension est basée sur l'utilisation d'un modèle SED à temps discret, appelé TDES (Timed Discrete-event system). Le modèle TDES est muni d'un événement spécial modélisant le temps, appelé *tick* d'horloge, représenté par le symbole τ. L'événement τ est supposé être observable. Le diagnostiqueur construit à partir de ce modèle estime l'état courant du système après l'occurrence d'un événement observable, ou un tick d'horloge τ.

Malgré les solutions de réduction proposées, le nombre d'états du diagnostiqueur reste assez important dû à l'explosion combinatoire causée par l'intégration de l'information temporelle. En effet, l'application de cette approche est confrontée en premier lieu, à la taille importante du modèle du système et celui du diagnostiqueur, et en second lieu à la complexité élevée de l'algorithme de synthèse du diagnostiqueur.

3.3.2.1.2 Approches à base de RdP temporel Dans (Ghazel et al., 2005), les auteurs ont proposé la construction d'un diagnostiqueur à partir d'un modèle RdP temporel du système. Cette approche exploite les contraintes temporelles sur les événements dans le but d'affiner les résultats de l'estimation. La construction du diagnostiqueur repose sur une approche d'analyse des états accessibles dans le RdP temporel du système. Le diagnostiqueur conséquent est un graphe d'états similaire au graphe des classes d'états, où chaque transition entre deux nœuds du graphe est étiquetée par un événement observable et un intervalle de franchissement. Une transition ne peut être franchie que si la date d'occurrence de l'événement observé satisfait l'intervalle de franchissement. Ainsi, suite à l'observation de chaque événement, une transition est franchie d'une manière déterministe vers un autre nœud du graphe, qui détermine la classe d'états accessibles à cette date, en tenant compte de l'occurrence de toute séquence d'événements non observables.

3.3.2.2 Approches en-ligne

Dans le cadre des approches en-ligne, aucun calcul pré établi n'est effectué. En effet, le diagnostiqueur élabore en-ligne le calcul nécessaire pour l'identification des défauts, à partir de l'ensemble d'événements observables générés par le système. Nous présentons dans la suite quelques travaux portant sur le diagnostic en-ligne qui sont basés sur des modèles SED temporisés.

3.3.2.2.1 Estimateur d'état de Tripakis Dans (Tripakis, 2002), Tripakis propose une extension temporisée de l'approche de diagnostic de Sampath reposant sur l'utilisation d'un estimateur d'état en-ligne (à ne pas confondre avec l'estimateur d'état défini par les automaticiens). La démarche de diagnostic présentée dans ce travail commence par l'élaboration d'un modèle automate temporisé du système à diagnostiquer. Par la suite, une partition des sommets de l'automate en sommets de fonctionnement normal et sommets de fonctionnement de défaut est établie. Le diagnostiqueur est donné sous la forme d'un algorithme d'estimation d'état en-ligne. Cet algorithme estime l'état courant du système suite à chaque occurrence d'un événement observable et déclenche une alarme quand un défaut est détecté.

Le fonctionnement du diagnostiqueur est décrit dans Algorithme 1.

Algorithm 1 Pseudo-code du diagnostiqueur

Initialiser W par l'état initial du système
Initialiser le temporisateur x à 0
boucler
  si $(f_d(W) = oui)$ **alors**
    Déclencher une alarme annonçant la détection d'un défaut
  finsi
  Attendre un événement $\sigma \in \Sigma_o$ **ou** l'expiration du temporisateur $(x = TO)$
  si un événement σ est observé **alors**
    Estimer l'état courant W suite à l'écoulement de x u.t. et l'occurrence de σ
  sinon
    Estimer l'état courant W suite à l'écoulement de x u.t.
  finsi
  Initialiser le temporisateur x à 0
fin boucle

Le temporisateur x permet de mesurer le temps écoulé depuis la dernière estimation effectuée (la date du dernier événement observé). L'estimation de l'état courant du système est donnée par l'ensemble W. Cette estimation comporte un ensemble d'états vers lesquels le système peut évoluer, en accord avec l'ensemble des événements observés. Nous rappelons qu'un état du système correspond à un sommet de l'automate temporisé associé aux valeurs prises par les horloges à un instant donné.

Lorsque le diagnostiqueur observe un événement σ, ou lorsque le temporisateur x atteint une valeur seuil notée TO $(x = TO)$, l'algorithme effectue une estimation de l'état courant du système à la date t_i. Il s'agit de calculer l'ensemble des états accessibles :

- à partir des états estimés à la date t_{i-1} (dernière estimation effectuée) ;
- suite à l'écoulement de $x = t_i - t_{i-1}$ u.t. et l'occurrence de toute possible séquence d'événements non observables ;

- enfin, l'occurrence de l'événement σ, si cet événement a été observé.

La fonction de diagnostic f_d évalue l'estimation de l'état courant du système et déclenche une alarme si elle détecte un défaut ; i.e, lorsque tous les états estimés dans W correspondent à des sommets de fonctionnement de défaut.

Un exemple d'application de l'algorithme du diagnostiqueur est illustré dans la figure 3.7. Nous supposons, dans cet exemple, une évolution du système décrite par la séquence d'événements temporisés : $(a, 9)(f, 3.5)(b, 1)(c, 4)$, où a, b et c représentent les seuls événements observables et f désigne l'événement de défaut. Après l'observation de la projection observable de cette séquence d'événements, à savoir, la séquence $(a, 9)(b, 4.5)(c, 4)$, l'algorithme du diagnostiqueur évolue comme décrit dans la figure 3.7.

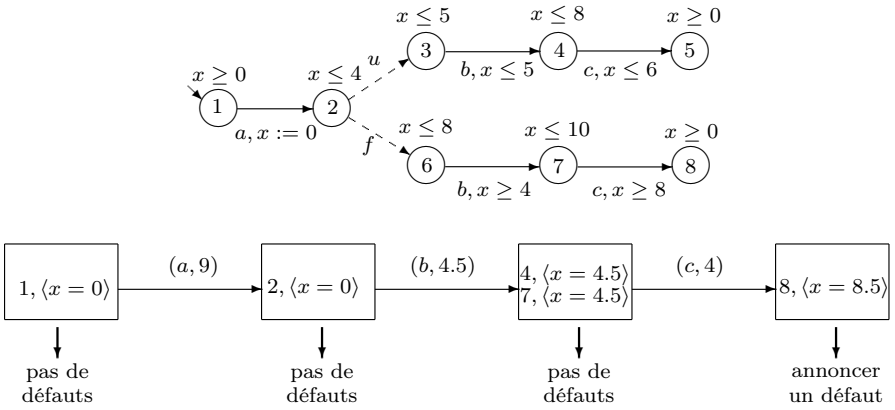

Fig. 3.7 – Une exécution de l'algorithme du diagnostiqueur

Dans cet exemple, l'occurrence de l'événement c nous a permis d'identifier le défaut f. Avant le franchissement d'une transition sur l'événement c, la valeur courante de l'horloge x est égale à 8.5, et l'état discret estimé du système correspond au sommet 4 ou au sommet 7. La transition sur l'événement c issue du sommet 4 n'est pas possible à $x = 8.5$, puisque la condition de garde $x \leq 6$ n'est pas satisfaite. Par contre, une transition à partir du sommet 7 sur l'événement c est possible puisque sa condition de garde $x \geq 8$ est satisfaite par la valeur de l'horloge x.

Comme il est le cas pour l'approche de Sampath, l'utilisation du diagnostiqueur dans cette approche suppose que le modèle automate temporisé du système est diagnosticable. Dans le cas échéant, le diagnostiqueur peut rester indéfiniment incapable de détecter l'occurrence d'un défaut. Intuitivement, il existe une paire de trajectoires temporisées du système ayant le même comportement observable, mais l'une contient un défaut et l'autre non. Dans la section 3.6, nous présentons la notion de diagnosticabilité dans le cadre des automates temporisés ainsi que les techniques utilisées pour sa vérification.

L'algorithme proposé par Tripakis est basé sur une estimation d'état dans un automate temporisé avec ϵ-transitions. L'estimateur d'état en-ligne se distingue par sa

simplicité et permet d'éviter le problème d'explosion combinatoire lié à la compilation hors-ligne du diagnostiqueur, cependant, il présente plusieurs inconvenants.

D'abord, la complexité d'exécution de cet algorithme est doublement exponentielle en fonction de la taille du modèle considéré et la taille de la séquence d'événements observée (Bouyer et Chevalier, 2005). Cette **grande complexité** du calcul en-ligne implique un phénomène de retard considérable, qui augmente au fur et à mesure du nombre d'événements observé (puisque la taille des observations intervient dans la formule de complexité). Ce retard n'est pas toléré lors du diagnostic des systèmes critiques, qui nécessitent une identification rapide des défauts, et peut avoir dans certains cas des conséquences néfastes. Nous remarquons que cette approche s'intéresse seulement à la **détection** et non à l'identification des défauts puisqu'elle considère l'existence d'un seul type de défaut. Enfin, cette approche est incapable de **détecter au plus-tôt** les défauts, quand le système se bloque et ne génère plus d'événements. En effet, il faut attendre l'écoulement d'une **durée constante** TO u.t. pour déclencher une alarme, même s'il est possible de s'assurer, plus-tôt, de l'occurrence du défaut.

3.3.2.3 Diagnostic à base de model-checking

Ces approches reposent sur l'utilisation d'une technique de vérification d'automates appelée *model-cheking* (Yovine, 1998). Cette technique se base sur des langages de spécification de propriétés tels que le PCTL, CTL ou TCTL, et un algorithme de vérification de propriétés dit model-checker. Le langage de spécification, basé sur une logique temporelle, permet de formuler des propriétés telles que l'atteignabilité, la vivacité,....

Dans (Cordier et al., 2001), les auteurs proposent une approche de diagnostic en-ligne basée sur la technique de model-checking et le langage de spécification TCTL. Cette approche repose, d'abord, sur une modélisation des composants du système sous la forme d'automates à états finis. Un ensemble d'horloges et un ensemble de contraintes temporelles seront ensuite introduits sur la composition de ces modèles élémentaires. L'automate temporisé résultant permet de prendre en considération l'aspect temporel correspondant à l'évolution du système. Ensuite, les auteurs proposent l'utilisation du model-checker Kronos (Yovine, 1997), afin d'expliquer, en-ligne, l'ensemble des observations générées par le système. Il s'agit de retrouver l'ensemble des trajectoires du système qui expliquent la trajectoire observable générée, à travers l'application d'une formule d'atteignabilité en utilisant le model-checker. Une interprétation des trajectoires retrouvées permet l'identification de défauts produits.

3.3.2.3.1 Diagnostic à base de chroniques Une chronique peut être considérée comme un ensemble de motifs d'événements liés entre eux par des relations de contraintes temporelles (Dousson, 1994). La méthode de diagnostic à base de chroniques associe à

chaque mode de défaillance, que l'on désire identifier, une chronique (ou un scénario). Ensuite, elle utilise une technique de reconnaissance en-ligne de ces chroniques afin d'identifier les scénarios de défaillances. En effet, si des événements observés correspondent aux motifs de la chronique et si leur occurrence a lieu selon le contexte et les contraintes spécifiées, alors une instance de la chronique modélisée devra être reconnue.

Comme il est illustré dans la figure 3.8, introduite dans (Dousson, 1994), les modèles de chroniques sont spécifiés hors-ligne alors que la phase de reconnaissance est établie en-ligne. Cette dernière phase repose sur le maintien à jour de fenêtres temporelles précisant les dates où un événement est attendu par la chronique compte-tenu de l'ensemble de ses contraintes. Les chroniques peuvent être modélisées par un RdP temporel ou par un graphe de contrainte (treillis) (Mokhtari, 2007).

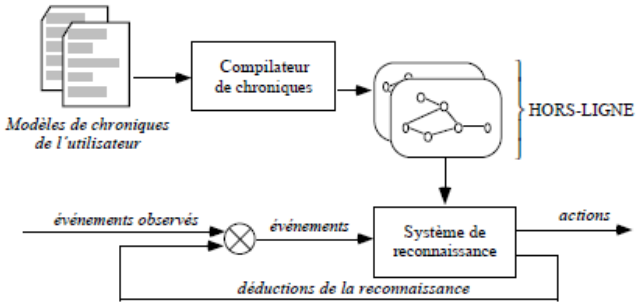

Fig. 3.8 – Un système de reconnaissance de chronique.

3.4 Du diagnostic des systèmes à événements discrets vers le diagnostic des systèmes dynamiques hybrides

La plupart des SAP réels évoluent selon des sous processus continus qui sont démarrés, arrêtés par des commandes à états discrets (dont les entrées dépendent des sous processus continus). Par conséquent, les procédés ont rarement un comportement purement discret ou purement continus mais plutôt un mélange entre les deux. Ces systèmes dynamiques à double composante comportementale (dynamique continue et événementielle) sont nommés : *Systèmes Dynamiques Hybrides* (SDH). Ces systèmes peuvent être de natures très diverses. On peut rencontrer des systèmes continus auxquels sont associés des commutations discrètes ou bien des systèmes à événements discrets auxquels sont associés certaines évolutions continues. Plusieurs outils de modélisation des systèmes hybrides ont été proposés dans la littérature. Parmi ces outils, nous citons les automates hybrides (Alur et al., 1993), les automates hybrides rectangulaires (Henzinger et al.,

1998), les automates hybrides linéaires (Müller et Stauner, 2000), les RdP hybrides (Alla et David, 1998; David et Alla, 2004), les statecharts hybrides (Harel et Pnueli, 1987), les bond graphs hybrides (Mosterman, 1997)... Dans la suite, nous insistons sur les outils dans lesquels nous avons trouvé des réponses aux objectifs que nous nous sommes fixés.

3.4.1 Modélisation des SDH

D'une manière générale, un SHD est modélisé par un ensemble de systèmes à dynamique continue interagissant avec un ou plusieurs systèmes à événements discrets (Kurovszky, 2002). Le point commun entre ces formalismes est que l'évolution continue est affectée par les événements discrets et les modèles nécessitent à la fois des variables d'état continues et discrètes.

Nous pouvons classer les approches de modélisation des SDH selon trois classes(Chen et Provan, 1997) :

- **l'approche continue** : cette approche consiste à définir une approximation des dynamiques discrètes du système hybride par des équations différentielles pour modéliser l'occurrence des événements discrets ;
- **l'approche événementielle** : avoir une approche purement discrète pour modéliser les SDH consiste à supprimer les dynamiques continues ou à faire une approximation de l'évolution continue de façon à ce que le système hybride soit représenté uniquement par les événements qui le caractérisent ;
- **l'approche mixte** : dans une approche mixte, chacune des deux composantes (discrète et continue) est représentée de façon rigoureuse et explicite et leur collaboration est prise en compte dans l'interface qui les relie. La résolution du modèle continu déclenche l'évolution des variables au cours du temps et valide certaines transitions. L'évolution du modèle discret engendre alors la mise en place d'un nouvel état discret qui se traduit par l'élaboration d'un nouveau système d'équations.

Nous présentons dans la suite quelques modèles mixtes rencontrés dans la littérature.

3.4.1.1 Les automates hybrides

Les automates hybrides (Alur et al., 1993) sont une extension des automates à états finis. Ils représentent des systèmes qui intègrent deux composantes : celle ayant un comportement discret, modélisée naturellement par un automate à états finis et celle dont le comportement varie de manière continue dans le temps, modélisée par un système algébrodifférentiel. Un automate hybride évolue par une alternance de pas continus, où les variables d'état et le temps évoluent de façon continue, et de pas discrets où plusieurs transitions discrètes et instantanées peuvent être franchies. Notons qu'un automate tem-

porisé correspond à automate hybride particulier où toutes les variables admettent des dérivées par rapport au temps égales à 1.

D'un point de vue informel et général, un automate hybride apparaît ainsi comme un automate à états fini pilotant un ensemble d'équations différentielles modélisant la dynamique continue du système. L'état de l'automate change instantanément lors de l'occurrence d'un événement discret ou par l'écoulement du temps lors de la validation d'une condition logique spécifiée sur la valeur de la variable continue (Henzinger, 1996).

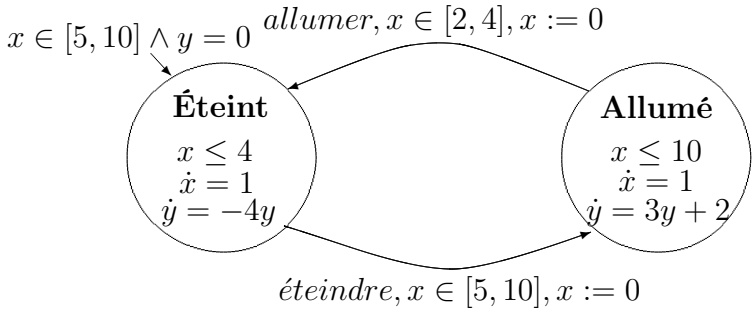

Fig. 3.9 – Automate hybride

Considérons l'automate représenté dans la figure 3.9 modélisant un système hybride. Dans ce modèle, l'évolution continue est représentée par des équations différentielles associées aux sommets du graphe et l'évolution événementielle est modélisée par les arcs étiquetés du graphe. Les sommets **Éteint** et **Allumé** représentent les états discrets du système où l'évolution continue a lieu. Les prédicats $x \in [2,4]$ et $x \in [5,10]$ sur les arcs traduisent les conditions, dites aussi *gardes*, pour l'occurrence des événements *allumer* et *éteindre*, respectivement. Les prédicats $x \leq 4$ et $x \leq 10$ dans les sommets représentent les *invariants* de l'automate, c'est-à-dire, des conditions imposées aux variables continues du système pour rester dans un état discret (ici les états **Éteint** ou **Allumé**). L'état initial du système est représenté par un arc d'entrée dans le sommet d'origine. L'étiquette de cet arc $x \in [5,10] \wedge y = 0$ représente la région de l'espace continue à partir de laquelle la dynamique du système hybride démarre. Les variables x et y évoluent dans le sommet **Éteint**, respectivement le sommet **Allumé**, conformément aux équations différentielles, $\dot{y} = -4y$ et $\dot{x} = 1$, respectivement $\dot{y} = 3y + 2$ et $\dot{x} = 1$, appelées *conditions de flux*[1].

L'analyse et la vérification des systèmes en utilisant les automates hybrides représente un thème central dans le cadre de l'étude des SDH. Il s'agit de vérifier quelques propriétés qualitatives et quantitatives sur un système en utilisant des méthodes automatiques. La vérification repose, d'abord, sur une modélisation du comportement du SDH à vérifier par un automate hybride. Ensuite, l'application d'une analyse des propriétés qualitatives à travers les techniques de model-checking (Henzinger et al., 1997), ou les propriétés quantitatives à travers l'analyse d'accessibilité (Alur et al., 1993, 1995).

[1]Nous désignons par $\dot{x}$ la dérivée du premier ordre par rapport au temps de la variable x.

L'analyse d'accessibilité consiste à déterminer l'espace d'état accessible par l'évolution du système hybride étudié. Ce problème n'est pas décidable pour un automate hybride sans hypothèses particulières (Henzinger et al., 1998). Il faut alors apporter des restrictions pour avoir des sous-classes pour lesquelles certaines propriétés sont décidables. Dans la suite, nous nous intéressons à l'étude d'une de ces sous-classes d'automates hybrides pour laquelle l'analyse d'accessibilité est décidable.

3.4.1.2 Les sous-classes d'automates hybrides

Plusieurs sous-classes du modèle automate hybride ont été explorées dans la littérature. Ces sous-classes ont été étudiées dans le but d'alléger la structure du modèle initial, afin de simplifier son analyse et sa vérification. Parmi les modèles proposés, nous citons :

- les automates hybrides linéaires (Alur et al., 1993) : un automate hybride est dit *linéaire* si les conditions de flux, des invariants, des gardes, sont définies par des expressions linéaires sur l'ensemble des variables.
- les automates hybrides rectangulaires (Kopke, 1996; Henzinger et al., 1998) : c'est une sous-classe des automates hybrides linéaires. La condition de flux dans ce modèle est définie sous la forme de prédicats rectangulaires de la forme $\dot{x} \in [a, b]$, pour chaque variable x du modèle. De même, Les invariants, les gardes, la condition initiale sont décrits par des prédicats rectangulaires.
- les automates hybrides rectangulaires initialisés (Henzinger et al., 1998) : c'est une sous-classe des automates hybrides rectangulaires. Dans ce modèle, chaque variable qui change de condition de flux, suite au franchissement d'une transition entre deux sommets, doit être réinitialisée. Dans la figure 3.10, nous illustrons l'exemple d'un automate hybride rectangulaire initialisé.

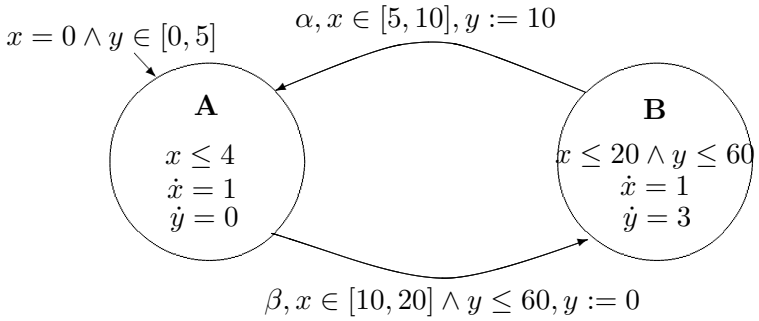

Fig. 3.10 – Automate hybride rectangulaire initialisé

La sous-classe des automates hybrides rectangulaires est une sous-classe très intéressante d'automates hybrides. Elle a été largement étudiée dans la littérature (Spathopoulos, 2000; Henzinger et Majumdar, 2000). L'importance de cette sous-classe est due au fait que plusieurs problèmes intéressants, tels que l'analyse d'accessbilité (Henzinger

et al., 1998), la synthèse de contrôleurs (Spathopoulos, 2000), le model-checking (Henzinger et Majumdar, 2000) sont décidables pour les automates hybrides rectangulaires initialisés. Ce modèle représente le modèle de base de notre travail de recherche. Une description plus détaillée et formelle de la sous-classe des automates hybrides rectangulaires sera donnée dans les chapitres suivants.

3.4.1.3 Réseaux de Petri hybrides

Les réseaux de Petri (Petri, 1962) ont été très utilisés comme outils de modélisation, analyse et synthèse pour les systèmes à événements discrets. Dans (Alla et David, 1998), les auteurs présentent une extension des réseaux de Petri (RdP), les réseaux de Petri hybrides. Le modèle RdP hybride hérite tous les avantages du modèle de réseaux de Petri tel que la représentation du parallélisme, de la synchronization et des conflits (David et Alla, 2004).

Un RdP hybride permet d'obtenir des modèles concis de systèmes réels. Il est ensuite possible de construire de manière algorithmique l'automate hybride correspondant (Allam et Alla, 1996) et d'appliquer tous les outils formels qui y sont développés.

3.5 Méthodes de diagnostic des SDH

L'efficacité d'une méthode de diagnostic de SDH peut se mesurer par sa capacité à exploiter, d'une manière optimale, les deux aspects présentés par ces système : l'aspect continu à travers les variables d'état continues et l'aspect discret à travers les événements discrets. Habituellement, ces deux aspects sont abordés par les approches différentes, issues des systèmes continus ou des systèmes à événements discrets (SED). En effet, la détection d'une anomalie sur une variable d'état continue est réalisée lorsque celle-ci dépasse un certain seuil, et sur une variable discrète lorsque l'occurrence d'un événement inattendu se produit, ou lorsqu'un événement attendu ne se produit pas.

Nous présentons dans cette section un survol des méthodes de diagnostic des SDH. Dans cette présentation, nous respectons une classification de ces méthodes selon : les méthodes issues des approches continues et les méthodes issues des approches SED.

3.5.1 Contributions fondées sur des approches continues

Généralement, ces méthodes se basent sur des approches de diagnostic issues de la communauté des systèmes continus, telles que la génération des résidus et les graphes causaux (Feng et al., 2000; Koutsoukos et al., 2001; Balluchi et al., 2002; Karsai et al.,

2003; Domlan et al., 2004). L'exploitation de l'aspect discret des SDH est généralement faible dans ces approches. Nous présentons dans la suite, d'une manière non exhaustive, quelques approches rencontrées dans la littérature, dans ce cadre.

3.5.1.1 Approches à base de techniques de génération de résidus

Plusieurs approches de diagnostic basées sur la génération de résidus ont été proposées dans la littérature. Nous pouvons citer à titre d'exemple l'approche de (Koutsoukos et al., 2001) . Dans cette approche, l'estimation en ligne du mode de fonctionnement des SDH est réalisée grâce à l'emploi de RdP. L'occurrence d'un défaut est détectée en comparant les grandeurs mesurées à celles attendues en tenant compte des commandes envoyées au processus. Le diagnostic de l'état du processus est ensuite réalisé à l'aide d'un arbre logique. Cette approche, adaptée aux SDH, présente l'avantage de détecter les anomalies dues aux variables continues et à l'occurrence d'événements perturbateurs.

Une autre technique de diagnostic de SDH a été proposée (Cocquempot et al., 2004). Cette technique repose sur les méthodes de diagnostic à base de redondance analytique. Le système à diagnostiquer est modélisé par un automate hybride. La détection de défauts est réalisée grâce à la génération de résidus entre les variables d'entrée et de sortie mesurées et les relations de redondance analytique déterminées à partir des entrées et des sorties ainsi que de leurs dérivées, indépendamment du mode discret du système. Le diagnostic des défaillances est réalisé à partir de résidus structurés spécifiques à chaque défaut. Dans (Balluchi et al., 2002), une solution basée sur l'utilisation d'un observateur hybride constitué d'un observateur continu et un observateur discret est proposée. L'observateur discret permet d'identifier l'état discret courant du système tandis que l'observateur continu estime l'évolution des variables continues.

3.5.1.2 Approche à base de raisonnement causal

Gomaa propose, dans (Gomaa et Gentil, 1996; Gomaa, 1997), une approche de diagnostic des SDH fondée sur une extension des RdP hybrides, dénommé *les réseaux de Petri continus causaux hybrides*($RdPC^2H$). Ce modèle intègre trois type de RdPs : un RdP continu temporisé qu'on a appelé Réseau de Petri Continu Causal hybride($RdPC^2H$) ; c'est un modèle approximatif modélisant de façon causale la partie continue du SDH ; un RdP classique modélisant le système de contrôle (SED) ; et un RdP classique modélisant l'interaction entre la partie continue et le système de contrôle. Les liens causaux (transitions) entre les variables continues sont représentés à travers des fonctions de transfert qualitatives basées sur les informations de gain, retard,.... L'auteur propose la conception d'un système de diagnostic de défauts basée sur un modèle $RdPC^2H$. Le système de détection de défauts, influençant les variables continues, est réalisé de façon asynchrone ;

la localisation de défauts est effectuée par le chaînage arrière/avant des liens causaux entre les variables en utilisant leurs informations temporelles.

Une autre approche reposant sur un raisonnement causal a été proposé dans (Karsai et al., 2003). Cette approche repose d'abord sur la modélisation du système par un modèle bond graph hybride (Mosterman, 1997) puis la génération d'un graphe de propagation des défauts, qui permet de décrire les relations causales et temporelles entre les différents modes de défauts d'un coté, et les observations associées d'un autre.

3.5.2 Contributions fondées sur des approches SED

Il existe peu de contributions de diagnostic pour les SDH, issues de méthodes SED. En effet, l'élaboration de telles méthodes se trouve en confrontation avec les dynamiques complexes et les indécidabilités liées aux SDH. Nous pouvons dégager deux grandes approches dans ce cadre, chacune détermine à sa manière, l'acquisition du modèle hybride.

La première approche est fondée sur l'abstraction des trajectoires continues et sa représentation à travers des modèles discrets dynamiques. En effet, elle consiste à établir des partitions qualitatives des grandeurs continues et de les représenter sous la forme d'états ou d'étiquettes de transition d'état. Cette approche représente une solution alternative aux approches fondées sur une représentation analytique précise. Parmi les approches développées dans ce cadre, nous citons les travaux de Bhowal (Bhowal et al., 2007) qui se base sur un modèle automate hybride à temps discret.

Une autre approche repose sur une acquisition expérimentale du modèle du système à travers l'abstraction des équations de l'espace d'état et les méthodes d'identification à base de réalisations expérimentales. Parmi les approches développées dans le cadre, nous citons les travaux de Lunze (Lunze, 2000, 2006).

3.5.2.1 Approche de diagnostic de Lunze

Lunze propose dans (Lunze, 2000, 2006), des algorithmes de diagnostic similaires au concept du diagnostiqueur de Sampath. Ainsi, l'inférence et l'évaluation des hypothèses sur les défauts sont effectuées à travers un algorithme de diagnostic, comme il est indiqué dans la figure 3.11. Le problème de diagnostic est associé à un problème d'observation d'état qualitatif. En effet, une abstraction qualitative des variables continues du système est effectuée à travers l'utilisation de quantificateurs. L'identification de défaut consiste alors à trouver le modèle f_i dont la trajectoire prédite est cohérente avec celle observée. La localisation est une conséquence de l'identification qui associe à chaque f_i un composant ou sous-système. Si la discrimination entre les défauts est faible alors une réévaluation de l'abstraction qualitative faite ou bien une réévaluation du schéma d'instrumentation.

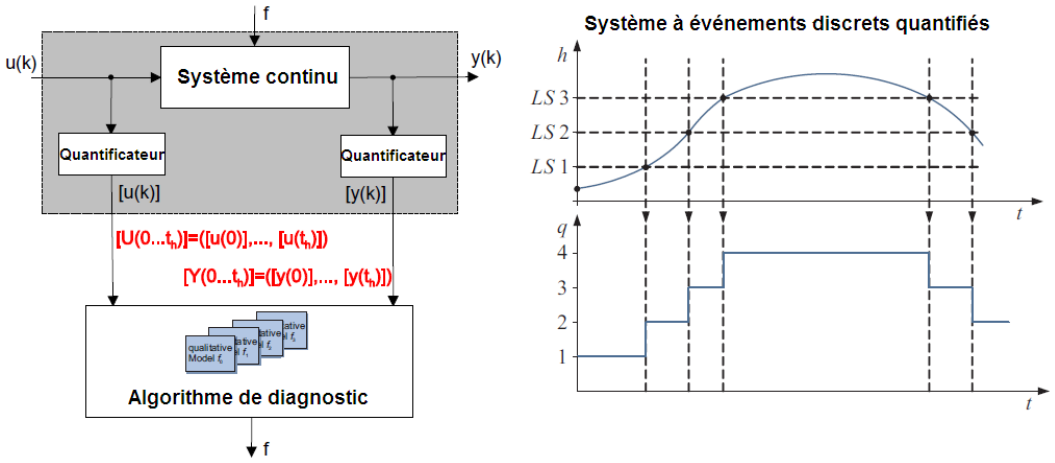

Fig. 3.11 – Principe de l'approche de diagnostic des SDH de Lunze

3.5.2.2 Approche à base de modèle hybride à temps discret

Une extension de l'approche de Sampath pour le diagnostic des systèmes hybrides à temps discret, a été proposée dans (Bhowal et al., 2007). Le modèle considéré évolue comme un automate temporisé à n différents flux (Alur et al., 1993). Il est représenté par un *graphe de transition d'activité*, où chaque activité correspond à un sommet du graphe. L'évolution des variables continues dans chaque activité est synchronisée sur l'occurrence d'un événement spécial, appelé *tick*, qui se produit après chaque période constante. Sur l'occurrence d'un tick, chaque variable évolue avec un pas constant, défini en fonction de sa dynamique. Les variables sont réparties en des variables discrètes (ou événements) observables et non observables et des variables continues mesurables et non mesurables.

L'approche de diagnostic commence par modéliser les composants du système en utilisant les graphes de transition d'activités, ensuite obtenir le modèle global par la composition de ces modèles. Le diagnostiqueur est un graphe de transition d'activité, compilé hors-ligne. Chaque sommet du diagnostiqueur contient une estimation de l'état du système et des étiquettes de l'ensemble $\{N, F_1, \ldots, F_m\}$ pour indiquer la présence de défauts.

Cette approche est basée sur l'abstraction des dynamiques continues et la discrétisation du temps. En effet, elle présente le même inconvénient que l'approche du diagnostic des SED à temps discret de Zad (Zad et al., 2005). Cet inconvénient correspond au problème d'explosion combinatoire de l'espace d'état du modèle utilisé, ainsi que celui du diagnostiqueur construit. Ce problème rend difficile la mise en oeuvre de cette approche pour les systèmes complexes.

3.6 Etude de la diagnosticabilité

Comme nous avons vu dans l'étude précédente, l'application d'une méthode de diagnostic est soumise à la condition de vérification du critère de diagnosticabilité. Ce critère stipule que le modèle du système dispose de suffisamment d'informations pour effectuer le diagnostic. En d'autres termes, il permet de répondre à l'interrogation suivante : est-il possible de déterminer toutes les occurrences des défauts affectant un système, étant données les informations fournies par son modèle et les observations qu'il génère ?

La définition de la diagnosticabilité pour les SED a été initialement formalisée dans les travaux de Lin et Wonham (Lin et Wonham, 1994). Plusieurs extensions de cette définition ont été ensuite proposées, dans de différents contextes (Sampath et al., 1995; Jiang et al., 2000; Tripakis, 2002; Fourlas et al., 2002; Thorsley et Teneketzis, 2005). Les extensions proposées varient en fonction de l'abstraction du modèle utilisé (logique, temporisé, hybride), de la structure de la méthode de diagnostic (centralisée,décentralisée, distribuée), ...

Dans la suite, nous présentons la notion de diagnosticabilité dans le cadre des SED logiques et temporisés. Nous nous intéressons aux travaux les plus importants pour la compréhension de notre travail. Ensuite, nous survolons les techniques utilisées pour la vérification de cette notion.

3.6.1 Diagnosticabilité des SED selon l'approche de Sampath

Pour une partition de défauts donnée, un SED est dit diagnosticable s'il est possible de détecter, au bout d'un délai fini, l'occurrence de n'importe quel défaut non observable, appartenant à un ensemble de la partition, à travers les événements observés (Lin et Wonham, 1994; Sampath et al., 1995). En effet, le critère de diagnosticabilité implique que chaque événement de défaut conduit à des observations suffisamment discriminantes pour permettre l'identification unique du mode de défaut au bout d'un délai fini. L'ensemble des observations doit être suffisamment riche pour cette discrimination. Cela peut être représenté formellement de la manière suivante :

Définition 7. Soient L un langage vivant préfixe-clos et Π_f une partition de défauts, Π_f=$\{\Sigma_{f_1}, \ldots, \Sigma_{f_m}\}$. L est dit *diagnosticable* par rapport à une partition de défauts Π_f, si :

$$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})(\forall s \in \Psi(\Sigma_{f_i}))(\forall t \in L/s)$$
$$||t|| \geq n_i \Rightarrow D$$

la condition de diagnosticabilité D est définie par :

$$\omega \in P_L^{-1}(P(st)) \Rightarrow \Sigma_{fi} \in \omega$$

où :

- L/s désigne l'ensemble des suffixes de L qui commencent par s, L/s=$\{t \in \Sigma^* \mid \text{st} \in L\}$.

- $P : \Sigma^* \rightarrow \Sigma_o^*$ est la fonction de projection sur l'ensemble des événements observables Σ_o. P permet de filtrer les événements non observables.

- P_L^{-1} désigne la fonction de projection inverse sur L définie par P_L^{-1}(y)=$\{\text{s} \in \text{L} \mid P(s) = y\}$.

- $\Psi(\Sigma_{fi})$ désigne l'ensemble des séquences terminées par un défaut de l'ensemble Σ_{f_i}.

Cette définition stipule qu'un langage L est diagnosticable si, pour toute séquence s contenant un défaut appartenant à l'ensemble Σ_{f_i}, le diagnostiqueur doit être capable d'identifier ce défaut après l'occurrence d'un nombre fini d'événements $n_i = ||t||$. En effet, toute autre séquence ω ayant un comportement observable que la séquence st ; i.e., $P(\omega) = P(st)$, elle doit contenir un défaut appartenant à Σ_{f_i}.

En effet, afin de pouvoir identifier un défaut de l'ensemble Σ_{f_i}, toute trajectoire observable qui suit l'occurrence du défaut, doit être distinguée, au bout d'un nombre fini d'événements, des trajectoires observables des comportements ne contenant pas de défauts de Σ_{f_i}. Autrement dit, l'occurrence d'un défaut de l'ensemble Σ_{f_i}, conduit à des observations suffisamment discriminantes pour permettre l'identification unique du type de défaut Σ_{f_i} dans un délai fini. Cet ensemble d'observation définit alors la signature du défaut.

Une approche systématique pour la vérification de la diagnosticabilité d'un language L, généré par un automate à états finis G, a été proposée dans (Sampath et al., 1995). Cette approche repose sur la construction du diagnostiqueur D, puis la vérification de certains cycles, dit F_i-indéterminés. Un cycle F_i-indéterminé est constitué exclusivement d'états F_i-incertain, à qui correspond deux cycles dans l'automate G, où tous les éléments du premier cycle sont associés à l'étiquette F_i dans le cycle du diagnostiqueur, et tous les éléments du deuxième cycle sont associés à des étiquettes différentes de F_i.

Un language L, généré par un automate G, est diagnosticable par rapport à une partition de défauts Π_f, si son diagnostiqueur D ne contient aucun cycle F_i-indéterminé, pour tout $i \in \Pi_f$. Considérons le diagnostiqueur construit dans l'exemple 3.3.1, ce diagnostiqueur ne vérifie pas la condition de diagnosticabilité, puisque le cycle constitué par

les états $\{3F_1\ 7N\}$, $\{4F_1\ 9F_2\ 12N\}$ et $\{5F_1\ 10F_2\ 13N\}$, est F_1-indéterminé. A ce cycle, correspond deux cycles d'états dans l'automate G : chaque élément dans le cycle des états $\{7, 12, 13\}$, est associé à l'étiquette F_1, et les éléments du cycle $\{3, 4, 5\}$, sont associés à des étiquettes différentes de F_1. Si le système génère la séquence $af_1bcdbcd(bcd)\ldots$, le diagnostiqueur va évoluer dans le cycle F_1-indéterminé, sans pouvoir identifier le défaut. Autrement dit, la séquence observée par le diagnostiqueur $abcdbcd(bcd)\ldots$, peut provenir d'une évolution du système dans le cycle $\{3, 4, 5\}$ avec l'occurrence d'un défaut $f_1 \in \Sigma_{f_1}$, soit d'une évolution du système dans le cycle $\{7, 12, 13\}$. L'auteur a proposé plusieurs solutions permettant de rendre le système diagnosticable : la première solution consiste à introduire des capteurs supplémentaires afin que l'ensemble des observations soit suffisamment riche pour permettre la discrimination du comportement de défaut et rendre ainsi le système diagnosticable. La deuxième solution, appelée diagnostic actif, consiste à restreindre certains comportements du système afin de le rendre diagnosticable. Cette méthode est basée sur la théorie de commande supervisée introduite par Ramadge-Wonham (Ramadge et Wonham, 1987). Une autre solution consiste à redéfinir la partition des défauts. En effet, si deux défauts, appartenant à deux différents ensembles de défauts, ne peuvent pas être distingués selon leurs trajectoires observables, alors on fusionne les deux ensembles de défauts en un même ensemble. Cependant, cette solution rend la capacité de localisation du diagnostiqueur plus faible.

3.6.2 Diagnosticabilité des SED à aspect temporel

Dans (Tripakis, 2002), l'auteur propose une extension de la notion de diagnosabilité, intégrant le temps, dite la Δ-*diagnosticabilité*. En effet, un automate temporisé est dit Δ-diagnosticable, s'il est possible de détecter tout défaut non observable, au bout d'un délai Δ u.t. après son occurrence. Il faut souligner que cette définition traite uniquement le problème de détection dans le sens où elle considère l'existence d'un seul type de défauts.

Une trace temporisée ω dite Δ-défaillante, si elle contient un défaut et au moins Δ u.t. se sont écoulées après l'occurrence de ce défaut. On définit l'opérateur de projection observable P qui permet de filtrer les événements non observables à partir d'une trace temporisée. Formellement, la Δ-diagnosticabilité est définie comme suit (Tripakis, 2002) :

Définition 8. Un langage temporisé L est dit Δ-*diagnosticable*, pour $\Delta \in \mathbb{N}$, si pour toute paire de traces temporisées ω et ω' dans L, si ω est Δ-défaillante alors ω contient un défaut ou $P(\omega) \neq P(\omega')$.

Cette définition stipule qu'un langage temporisé est Δ-diagnosticable, si toute paire de traces dans L, la première contient un défaut et la seconde est dépourvue de défauts, les projections observables de ces deux traces doivent être différentes au bout de Δ u.t. de l'occurrence du défaut.

L'intégration du temps dans la définition de Δ-diagnosticabilité introduit deux nouvelles notions sur la définition classique de diagnosticabilité :

- dans l'approche classique, l'identification d'un défaut doit se faire au bout d'un délai fini. Ce délai est représenté qualitativement à travers le nombre d'événements qui suivent l'événement du défaut. Par contre, ce délai est défini quantitativement dans l'extension temporisée de cette définition à travers le délai Δ.
- la discrimination des trajectoires de défauts prend en considération le temps d'une manière explicite. En effet, la définition classique stipule qu'un language est diagnosticable s'il est possible de discriminer, d'un point de vue logique, les trajectoires comportant des défauts. Cela nécessite l'observation d'un **événement discriminant**, au bout d'un délai fini, permettant de caractériser d'une manière unique une trajectoire défaillante du système. Dans le contexte temporisé, la discrimination d'une trajectoire défaillante considère en plus, les dates d'occurrence des événements observables. Ainsi, deux traces peuvent avoir la même projection observable d'un point de vue logique (séquence d'événements) mais pas d'un point de vue temporisé (dates des événements). Ainsi, il est possible de discriminer un comportement défaillant en s'appuyant sur les dates d'occurrence des correspondants événements observables. Dans ce cas, on parle de **temps discriminant**.

Exemple 3.6.1
Pour une meilleure compréhension de la notion de Δ-diagnosticabilité, nous considérons l'exemple suivant, introduit dans (Tripakis, 2002). Nous supposons que les événements a et b sont observables, les événements u et f non observables et f l'événement de défaut. Il est clair que le langage temporisé accepté par l'automate temporisé A illustré dans la figure 3.12 est 3-diagnosticable. D'abord, il faut noter que dans le comportement défaillant, le délai entre l'événement b et l'événement de défaut est inférieur à 3 u.t., ce qui implique l'existence de l'événement b dans toute trace 3-défaillante. L'observation de l'événement b nous permet de détecter le défaut dans tout comportement défaillant. En effet, dans chaque comportement défaillant, le délai entre les événements observés a et b est supérieur à 3 u.t.., tandis que ce délai est inférieur à 3 u.t. dans chaque comportement normal.

Si nous considérons dans un second exemple l'automate temporisé A' illustré dans la figure 3.13. Le langage accepté par cet automate n'est pas diagnosticable. En effet, si nous considérons les deux traces $(a,0)(f,2.5)(b,0.1)$ et $(a,0)(u,2.5)(b,0.1)$ acceptées par A', nous pouvons établir qu'elles ont la même projection observable ; i.e., $(a,0)(b,2.6)$, que la première trace contient un défaut et que la seconde ne contient aucun défaut. En plus, pour toutes continuations de ces deux traces obtenues par l'écoulement du temps, elles admettent la même projection observable.

L'auteur propose une technique de vérification de la diagnosticabilité, reposant sur

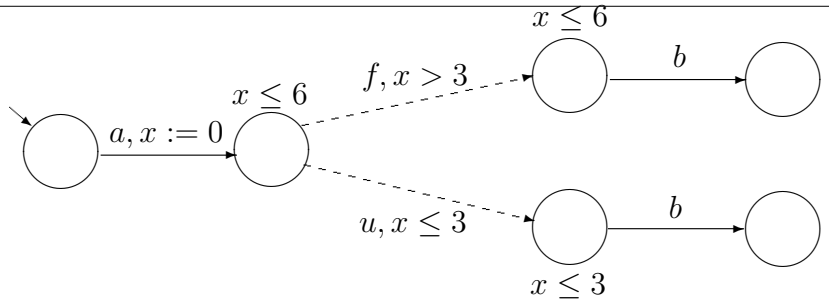

Fig. 3.12 – Automate temporisé 3-diagnosticable

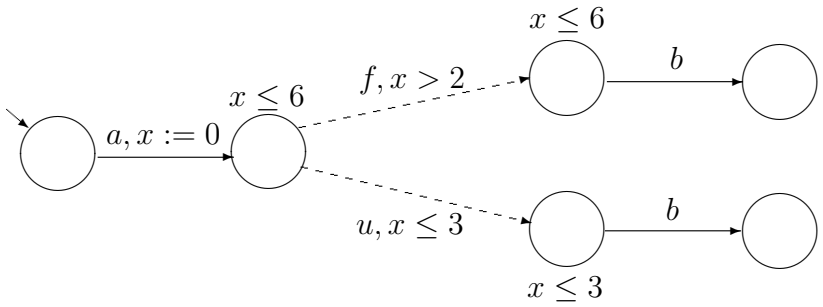

Fig. 3.13 – Automate temporisé non diagnosticable

la construction d'un produit spécial de deux versions modifiées du modèle du système, ensuite, la vérification de l'existence des trajectoires non-zenon (Alur et Dill, 1994) dans ce produit. Intuitivement, l'existence d'une trajectoire non-zenon dans ce produit spécial, implique l'existence de deux traces infinies du système, admettant la même projection observable, la première correspond à un comportement défaillant et la seconde à un comportement normal.

3.7 Conclusion

Dans ce chapitre, nous avons proposé un tour d'horizon sur les méthodes de surveillance et diagnostic à base de modèles des systèmes à événements discrets et des systèmes dynamiques hybrides. Nous avons classifié ces méthodes selon différents critères, parmi lesquels, le modèle utilisé pour représenter le comportement du système, la représentation du temps : modèles logiques, denses, à temps discret. Nous énumérons quelques remarques retenues à partir de notre étude :

1. une approche de référence a été proposée dans (Sampath et al., 1995), pour le diagnostic des SED :
   - cette méthode repose sur la conception d'un modèle complet du système décrivant le comportement normal et les comportements défaillants. Les événements

de défauts sont représentés par des événements non observables ;
- un diagnostiqueur, sous la forme d'un automate à états finis, est construit hors-ligne, à partir du modèle du système. Ce diagnostiqueur est par la suite utilisé en-ligne, pour inférer l'occurrence des événements de défaut en observant une séquence d'événements observables générée par le système ;
- une notion de diagnosticabilité a été définie. Un modèle diagnosticable garantit la discrimination de tout défaut au bout d'un nombre fini de transitions après l'occurrence du défaut ;
- le diagnostic élaboré par cette approche considère une évolution complètement discrète du système. En effet, le temps est considéré d'une manière qualitative à travers l'ordre d'occurrence des événements ;

2. l'exploration des données temporelles relatives aux événements qui peuvent avoir lieu dans un SED s'est avérée très fructueuse dans un processus de surveillance et du diagnostic. En effet, elle augmente la capacité d'identification des défauts du diagnostiqueur ;
3. une extension temporisée de l'approche de Sampath, reposant sur un modèle SED à temps discret, a été proposée dans (Zad et al., 2003). Cette approche utilise un événement spécial appelé tick d'horloge, pour représenter le passage du temps. La discrétisation du temps entraîne une explosion combinatoire de la taille du modèle du système ainsi que celui du diagnostiqueur. Ainsi, cette approche est inadéquate pour le diagnostic des systèmes complexes ;
4. une extension inspirée des travaux de Sampath, dans le contexte des modèles temporisés denses, a été proposée dans (Tripakis, 2002) :
   - cette approche repose sur l'utilisation d'un algorithme d'estimation d'état du système, exécuté en-ligne avec le système à diagnostiquer ;
   - une extension de la définition de diagnosticabilité a été également proposée dans le cadre des langages temporisés ;
   - la complexité du diagnostic en-ligne est doublement exponentielle en fonction de la taille de l'observation et du modèle du système ;
5. l'élaboration d'un complexe calcul en-ligne pénalise les performances du diagnostic et le rend parfois l'approche inadéquate pour les systèmes temps réels critiques, qui nécessitent une prise de décision rapide de la part du diagnostiqueur ;
6. conception hors-ligne d'un diagnostiqueur sous la forme d'un automate temporisé est plus appropriée pour ces systèmes puisqu'elle nécessite peu d'effort en-ligne ;
7. la plupart des méthodes de diagnostic des SDH rencontrées dans la littérature sont issues de contributions du domaine du diagnostic des systèmes continus ;
8. les méthodes de diagnostic issues de contributions SED, sont des méthodes qualitatives qui se basent sur une abstraction discrète de la dynamique continue et/ou du temps ;

9. à notre connaissance, il existe peu de contributions de diagnostic pour les SDH, reposant sur des modèles hybrides mixtes ;
10. à notre point de vue, l'indécidabilité du problème de vérification des modèles hybrides représente la principale difficulté empêchant le développement d'approches de diagnostic à base de modèles hybrides denses.
11. le problème d'analyse d'accessibilité est décidable pour certaines sous-classes de modèles hybrides, telles que les automates temporisés et les automates hybrides rectangulaires initialisés.

D'une manière générale, la principale difficulté du diagnostic à base de modèles des SED et des SDH, est de trouver un bon compromis entre le degré de précision de la représentation des systèmes et la taille et la complexité du modèle. Plus le modèle est fin, plus les possibilités d'identification de défauts augmentent et plus le modèle est grand et complexe, plus les besoins en temps et en mémoire sont grands. Toutefois, l'augmentation de la précision du modèle rend la solution proposée vulnérable par de nombreux problèmes d'indécidabilités. Dans ce cas, il faut considérer des sous-classes bien déterminées du modèle général.

Cet état de l'art nous a permis de situer notre approche de diagnostic pour les systèmes temporisés et les SDH, que nous allons développer dans les chapitres suivants. Il permet aussi de justifier les différents choix que nous allons établir dans le cadre de notre contribution.

Dans un premier volet, nous allons proposer une méthode de diagnostic pour les SED, basée sur un modèle automate temporisé du système, vérifiant certaines hypothèses. Notre approche repose sur la compilation hors-ligne d'un diagnostiqueur sous la forme d'un automate temporisé.

Dans un deuxième volet, nous proposons un algorithme d'estimation d'état pour le diagnostic des SDH. Cet algorithme estime à la volet l'état du système ainsi que les défauts produits. Dans nos approches, nous présentons une étude de la diagnosticabilité du modèle utilisé et nous donnons une approche permettant sa vérification.

Chapitre 4

Diagnostic des systèmes temporisés

4.1 Introduction

Les travaux de recherche menés dans le domaine du diagnostic des systèmes à événements discrets ont révélé l'importance du facteur temps. Plusieurs études ont montré que ce facteur est souvent un porteur potentiel d'informations dans le cadre de la détection et du diagnostic. L'exploration des données temporelles relatives aux événements occurrant dans un SED s'est avérée très fructueuse dans un processus de diagnostic.

C'est à partir de ces faits que nous élaborons, dans ce chapitre, une approche de diagnostic pour les Systèmes à Événements Discrets dont on connaît le comportement temporel. La démarche de notre approche de diagnostic commence par la conception d'un modèle temporisé du SED, décrivant son comportement normal ainsi que ses possibles comportements défaillants. Ce modèle est donné sous la forme d'un automate temporisé respectant certaines hypothèses. Le choix du modèle automate temporisé est justifié par la généralité de ce modèle, sa capacité d'analyse et sa description intrinsèque des contraintes temporelles existantes entre les différents événements du système (Bengtsson et Yi., 2004). Nous considérons par ailleurs que les événements qui peuvent avoir lieu sont de deux types : *observables* et *non observables*. Dans notre approche, nous visons à maximiser l'exploitation des informations temporelles formulées par le modèle du système en s'appuyant sur un ensemble d'outils, développés dans la littérature, permettant l'analyse des automates temporisés.

La deuxième étape de notre démarche de diagnostic repose sur la construction, hors-ligne, d'un automate temporisé, appelé *diagnostiqueur*, à partir du modèle du système. Le diagnostiqueur est par la suite déployé, en-ligne, pour permettre l'identification des défauts affectant le système. En effet, l'automate du diagnostiqueur reçoit les événements observables générés par le système et évolue d'une manière *déterministe* d'un sommet à un autre. Chaque sommet du diagnostiqueur fournit une estimation de l'état courant du

système ainsi que les possibles défaillances pouvant affecter son fonctionnement. En se basant sur une analyse des données présentes dans le sommet courant du diagnostiqueur, une fonction de décision émet une alarme lorsqu'un défaut est identifié.

Dans ce chapitre, nous commençons par définir le modèle automate temporisé qui constitue le cadre formel de notre approche de diagnostic à base de modèles temporisés. Ensuite, nous précisons le contexte général ainsi que les objectifs de la réalisation d'une telle approche. En effet, nous illustrons, à travers des exemples, les différents scénarios de discrimination de défaut qui doivent être pris en considération par notre approche. Par la suite, nous présentons notre démarche de diagnostic à base de modèles temporisés. D'abord, nous caractérisons les différentes hypothèses retenues sur le modèle du système à diagnostiquer. Ensuite, nous détaillons l'algorithme de construction du diagnostiqueur à partir de la sous-classe de modèles considérée. Dans la section suivante, nous présentons la notion de diagnosticabilité pour les langages temporisés ainsi qu'une méthode systématique permettant de vérifier la diagnosticabilité du modèle considéré dans notre travail. Enfin, nous exposons quelques éléments relatifs à l'implémentation et la complexité de notre contribution.

4.2 Les Automates Temporisés : présentation et analyse

Nous présentons dans cette section le modèle automate temporisé introduit par Alur et Dill dans (Alur et Dill, 1994). Nous faisons un rappel de l'aspect syntaxique et sémantique de ce formalisme de modélisation. Nous portons une attention particulière à l'analyse symbolique de ce modèle.

4.2.1 Les langages temporisés

Soit W un ensemble quelconque, alors W^* (respect. W^ω) désigne l'ensemble des séquences finies (respect. infinies) d'éléments de W. On note $W^\infty = W^* \cup W^\omega$. Nous désignons par $\mathbb{Z}$ l'ensemble des entiers relatifs, $\mathbb{R}_+$ l'ensemble des nombre réels positifs et $\mathbb{R}_+^n$ l'espace euclidien de dimension n sur $\mathbb{R}_+$.

Définition 9. Un *mot temporisé* (ou une *trace temporisée*) est une séquence de couples $(\sigma_i, d_i)_{i>0} \in (\Sigma \times \mathbb{R}_+)^\infty$, où le délai $d_{i+1} \in \mathbb{R}_+$ désigne le temps écoulé entre les événements σ_i et σ_{i+1}.

Étant donnée une trace temporisée $\omega = (\sigma_1, d_1)(\sigma_2, d_2) \ldots (\sigma_k, d_k) \ldots$, on définit l'opérateur $\| \omega \|$ qui permet de projeter les éléments d'une trace temporisée ω sur l'ensemble des événements Σ : $\| \omega \| = \sigma_1\ \sigma_2\ \ldots \sigma_k \ldots$.

Définition 10. Étant donnés deux ensembles d'événements Σ et Σ', avec $\Sigma' \subseteq \Sigma$. On définit l'opérateur $P_{\Sigma'} : (\Sigma \times \mathbb{R}_+)^\infty \rightarrow (\Sigma' \times \mathbb{R}_+)^\infty$ de projection sur Σ' qui permet d'éliminer d'une trace temporisée ω tous les événements n'appartenant pas à l'ensemble Σ', puis de mettre à jour les délais entre le restant des événements.

Par exemple, étant donnée une trace temporisée $\omega = (a,5)(b,3)(c,4)(d,8)(e,1)$, la trace $P_{\{a,c,e\}}(\omega) = (a,5)(c,7)(e,9)$ correspond à la projection de ω sur l'ensemble des événements $\Sigma' = \{a,c,e\}$. Nous remarquons qu'après avoir éliminé l'événement b de la trace ω, nous avons mis à jour le délai entre les événements a et c. En effet, le délai calculé entre les événements a et c (égale à 7) correspond à la somme du délai entre a et b (égale à 3) et celui entre b et c (égale à 4).

Définition 11. Un *langage temporisé* L est un ensemble de séquences temporisées ; i.e., $L \subseteq (\Sigma \times \mathbb{R}_+)^\infty$. Étant donné un ensemble d'événements Σ', on définit la projection du langage L sur l'ensemble Σ', notée par $P_{\Sigma'}(L)$, comme l'ensemble constitué par les projections sur Σ' de toutes les traces de L.

4.2.2 Définition d'un automate temporisé

Un automate temporisé est un automate à états finis muni d'un ensemble de variables réelles positives appelées *horloges*. Les horloges sont incrémentées simultanément, avec une dynamique égale à 1 et peuvent être remises à zéro. Les horloges sont des variables fictives dans le sens où elles ne sont pas des variables du système, mais elles sont utilisées pour mesurer le temps et définir les contraintes temporelles sur le franchissement des transitions.

Une transition, entre deux sommets d'un automate temporisé, est franchie suite à l'occurrence d'un événement en entrée et la satisfaction d'une contrainte de franchissement, appelée *garde*. Ce franchissement est instantané et peut déterminer la mise à zéro d'un ensemble d'horloges.

Avant de définir formellement le modèle automate temporisé, nous introduisons, dans ce qui suit, quelques notions préliminaires.

Définition 12. Étant donné un ensemble d'horloges X, l'ensemble des contraintes sur X, noté $\mathcal{C}(X)$, est défini par la grammaire :

$$\psi ::= x \sim c \mid x - y \sim c \mid \psi \wedge \psi \mid \neg\psi \mid vrai$$

où $x, y \in X$, $c \in \mathbb{Z}$, $\sim \in \{<, \leq, =, \geq, >\}$.

On notera le complément de la contrainte $vrai$ par le symbole $\emptyset$. Par exemple, on note $x < 3 \wedge x \geq 10 = \emptyset$

Définition 13. L'ensemble des gardes sur X, noté $\mathcal{G}(X)$, est le sous-ensemble de $\mathcal{C}(X)$, ne contenant pas des contraintes de la forme $x - y \sim c$, dites *contraintes diagonales*. En effet, l'ensemble $\mathcal{G}(X)$ est défini par la grammaire suivante :

$$g ::= x \sim c \mid g \wedge g \mid \neg g \mid vrai$$

où $x \in X$, $c \in \mathbb{Z}$, $\sim \in \{<, \leq, =, \geq, >\}$.

Définition 14. Un *automate temporisé* est défini par un septuplet $A = (Q, X, \Sigma, E, q_0, Q^f, I)$ (Alur et Dill, 1994) où,

- Q est un ensemble fini de sommets ;
- $X = \{x_1, \ldots, x_n\}$ est un ensemble fini d'horloges ;
- Σ est un ensemble fini d'événements (ou d'actions ou de symboles) ;
- E est une relation de transition entre les sommets ;
- $q_0 \in Q$ est le sommet initial ;
- $Q^f \subseteq Q$ est un sous-ensemble de sommets finaux qui n'admettent pas des transitions de sortie vers d'autres sommets (sommets puits) ;
- $I : Q \rightarrow \mathcal{G}(X)$ est une fonction qui associe pour chaque sommet $q \in Q$, une contrainte $I(q)$ dans $\mathcal{G}(X)$, appelée *invariant* du sommet q.

Chaque transition $e \in E$ est définie par un quintuplet $e = (q, \sigma, g, Y, q')$ (noté aussi par $q \xrightarrow{\sigma, g, Y} q'$), où :

- q et q' désignent, respectivement, le sommet de départ et le sommet d'arrivée de la transition ;
- $\sigma \in \Sigma$ correspond à l'événement déclenchant le franchissement de la transition ;
- $g \in \mathcal{G}(X)$ est une contrainte sur l'ensemble des horloges, appelée *garde* de franchissement. Cette contrainte doit être satisfaite par les valeurs des horloges pour permettre le franchissement de la transition ;
- $Y \subseteq X$ est un sous-ensemble d'horloges remises à zéro lors du franchissement de la transition, appelé l'*ensemble de réinitialisation*.

Définition 15. Un automate temporisé est dit *déterministe* si pour chaque paire de transitions (q, σ, g, Y, q') et (q, σ, g', Y', q'') ayant le même événement de franchissement σ et issues du même sommet q, les contraintes de garde g et g' sont disjointes ; i.e., l'ensemble des valuations d'horloges satisfaisant à la fois les deux gardes est vide.

Remarque 2. Pour des raisons de lisibilité, lorsque la garde d'une transition vaut *vrai* ou que l'ensemble de réinitialisation est vide, le champ de la composante en question ne sera pas représenté.

Dans la suite, nous présentons un exemple permettant d'illustrer la modélisation d'un système avec un automate temporisé.

Exemple 4.2.1

Nous présentons dans la figure 4.1-a un simple système de chauffage de liquides. Ce système comporte deux vannes V_1 and V_2, un bac, un thermostat et deux capteurs de niveau : le capteur S_1 qui surveille le niveau maximal et le capteur S_2 qui surveille le niveau minimal. Le système commence par une phase de remplissage, en utilisant la vanne V_1. Dès que le niveau du liquide atteint le niveau maximal, après une intervalle de [40,50] u.t., un capteur de niveau émet l'événement *rempli*, la vanne V_1 passe en position fermée et le système commence la phase de chauffage qui dure 60 u.t.. Ensuite, le contrôleur commande l'évacuation du liquide présent dans un bac en ouvrant la vanne V_2. La phase d'évacuation dure entre 20 et 25 u.t.. Elle s'achève lorsque le capteur de niveau S_2 détecte que bac est vide et alerte le contrôleur par l'émission de l'événement *vide*. A l'interception de cet événement, le contrôleur ferme la vanne V_2 pour commencer un nouveau cycle du système.

L'automate temporisé décrit dans la figure 4.1-b présente le fonctionnement de ce système. Il comporte trois sommets chacun correspond à une phase de fonctionnement du système. Le passage d'une phase à une autre se fait suite à l'occurrence d'un événement : *rempli*, *évacuer* ou *vide*. L'horloge x permet de mesurer le temps écoulé après l'occurrence de chacun de ces événements. En effet, cette horloge est remise à zéro après le franchissement d'une transition sur chacun de ces événements. Puisque les conditions des invariants sont toutes bornées, le système ne peut séjourner dans l'un de ces sommets que pendant une durée finie. Par exemple, pendant la phase du remplissage, le système ne peut pas séjourner plus que 50 u.t. dans le sommet **remplissage**. Ainsi, une transition sur l'événement *rempli*, vers le sommet **chauffage**, doit être franchie lorsque la valeur de x se trouve dans l'intervalle [40,50].

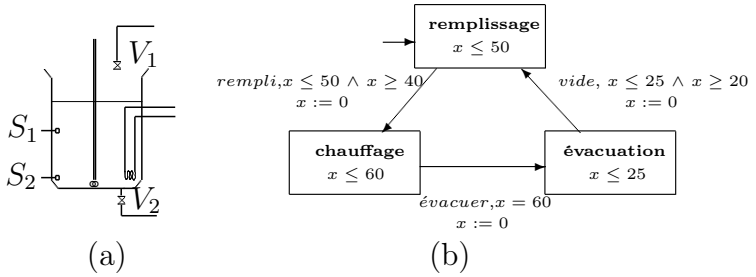

Fig. 4.1 – *Exemple d'un automate temporisé modélisant un système de chauffage de liquides.*

4.2.3 Sémantique

Définition 16. Soit X un ensemble fini d'horloges. Une *valuation* des horloges est une fonction $v : X \to \mathbb{R}_+$, qui associe un nombre réel positif à chaque horloge de X.

Étant donnée une valuation v sur X et un réel positif $d \in \mathbb{R}_+$, alors la valuation $v+d$ correspond à l'écoulement d'une durée de d unités de temps (u.t.) depuis la valuation v :

$$(v + d)(x) = v(x) + d, \forall x \in X$$

Soit $Y \subseteq X$ un sous-ensemble d'horloges de X, alors la valuation $v[Y \leftarrow 0]$ correspond à la valuation qui associe 0 pour toutes horloges dans Y, et $v(x)$ pour toutes horloges dans $X - Y$:

$$v[Y \leftarrow 0] = \begin{cases} 0, & x \in Y\,; \\ v(x), & x \in X - Y. \end{cases}$$

Définition 17. Un *état* d'un automate temporisé est un couple (q, v), où q est un sommet et v une valuation d'horloges. La configuration initiale d'un automate temporisé est (q_0, v_0) où $v_0(x) = 0$ pour toute horloge $x \in X$. La notation $v \vDash g$, où $v \in \mathbb{R}_+^n$ et $g \in \mathcal{G}(X)$, signifie que la valuation v vérifie la garde g.

Définition 18. Un *chemin* dans un automate temporisé A est une séquence finie ou infinie de transitions de E de la forme

$$p = q_0 \xrightarrow{\sigma_1, g_1, Y_1} q_1 \xrightarrow{\sigma_2, g_2, Y_2} q_2 \xrightarrow{\sigma_3, g_3, Y_3} q_3 \dots$$

où q_0 désigne le sommet initial.

D'une manière informelle, l'automate temporisé admet deux types d'évolution possibles :

- *les transitions de temps :* l'automate séjourne dans le même sommet en laissant le temps s'écouler. En effet, les valeurs des variables progressent d'une manière synchrone avec la même durée, en respectant la contrainte de l'invariant du sommet. Formellement, $(q, v) \xrightarrow{d} (q, v+d)$ pour $d \in \mathbb{R}_+$, si $\forall 0 \leq t \leq d$, $v + t \vDash I(q)$.

- *les transitions d'actions :* Ces transitions font évoluer l'automate d'un sommet à un autre. Une transition d'action est possible lorsque la valuation courante des horloges satisfait la contrainte de garde et un événement est réalisé. Ces transitions sont instantanées en plus. En plus, le franchissement d'une transition peut remettre à zéro un sous-ensemble d'horloges. Formellement, $(q, v) \xrightarrow{\sigma} (q', v')$ s'il existe $q \xrightarrow{\sigma, g, Y} q' \in E$ tel que $v \vDash g$, $v' = v[Y \leftarrow 0]$ et $v' \vDash I(q')$.

Définition 19. Une *exécution* r sur le chemin p est une suite de transitions d'actions et de transitions de temps définie de la manière suivante :

$$\langle q_0, v_0\rangle \xrightarrow[d_1]{\sigma_1,g_1,Y_1} \langle q_1, v_1\rangle \xrightarrow[d_2]{\sigma_2,g_2,Y_2} \langle q_2, v_2\rangle \xrightarrow[d_3]{\sigma_3,g_3,Y_3} \langle q_3, v_3\rangle \dots$$

où chaque délai $(d_i)_{i\geq 1}$ correspond à la durée de temps écoulée entre les événements σ_{i-1} et σ_i et $(v_i)_{i\geq 0}$ sont des valuations d'horloges définies comme suit :

- $v_0(x) = 0, \forall x \in X$,
- $\forall i \geq 1,\ v_{i-1} + d_i \vDash g_i$,
- $\forall i \geq 1,\ v_i(x) = \begin{cases} 0, & \text{si } x \in Y_i \\ v_{i-1}(x) + d_i, & \text{sinon.} \end{cases}$

Définition 20. Une séquence (ou une trace) temporisée $\omega = (\sigma_1, d_1)(\sigma_2, d_2)(\sigma_3, d_3)\dots$ est dite *acceptée* par un automate temporisé A, s'il existe une exécution de A qui évolue sur les éléments de ω.

Définition 21. L'ensemble des séquences qui sont acceptées par l'automate A forment le *langage accepté* (ou reconnu) par A, noté $L(A)$. En cas d'absence d'ambiguïtés, nous notons ce langage simplement par L.

Définition 22. Un état (q', v') d'un automate temporisé est dit *atteignable* (ou *accessible*) depuis un état (q, v), qu'on note par $(q, v) \rightsquigarrow (q', v')$, s'il existe une exécution qui commence à l'état (q, v) et qui progresse vers l'état (q', v'). Nous notons par $(q_0, v_0) \overset{\omega}{\rightsquigarrow} (q, v)$, un état (q, v) atteignable depuis l'état initial (q_0, v_0), à travers une exécution sur la trace temporisée ω.

Exemple 4.2.2

Considérons l'automate temporisé introduit dans l'exemple 4.2.1. Nous illustrons une exécution possible de l'automate de la figure 4.1-b, sur la trace temporisée $(rempli, 47.4)(évacuer, 60)(vide, 23.1)\dots$:

$$\langle \textbf{remplissage}, 0\rangle \xrightarrow[47.4]{rempli, x\leq 50 \wedge x\geq 40, x:=0} \langle \textbf{chauffage}, 0\rangle \xrightarrow[60]{évacuer, x=60, x:=0} \langle \textbf{évacuation}, 0\rangle \xrightarrow[23.1]{vide, x\leq 25 \wedge x\geq 20, x:=0} \langle \textbf{remplissage}, 0\rangle \dots$$

On définit l'opérateur *temps* qui, étant donnée une trace temporisée $\omega = (\sigma_1, d_1)(\sigma_2, d_2)(\sigma_3, d_3)\dots$, calcule (la limite de) la somme des délais $(d_i)_{i\geq 1}$; i.e., $temps(\omega) = \sum_{i\geq 1} d_i$.

Définition 23. Soit L un langage temporisé, alors :

- une trace infinie $\omega \in L$ est dite à *temps-divergent*, si $temps(\omega) = \infty$;
- une trace temporisée $\omega \in L$ est dite *admissible*, si elle correspond au préfixe d'une trace à temps-divergent de L ;

- le langage temporisé L est dit à *temps-divergent*, si toutes les traces temporisées du langage L sont admissibles ; i.e., toute trace infinie de L est à temps-divergent et toute trace finie est le préfixe d'une trace infinie à temps-divergent ;

La propriété de divergence du temps garantit l'absence des traces infinies qui s'exécutent dans un temps fini, dites aussi exécutions zénon (Tripakis, 1999).

Définition 24. Un automate temporisé A est dit *Fortement Non-Zénon* (FNZ) (Tripakis, 1999), s'il existe un entier naturel $d > 0$, tel que chaque exécution évoluant dans un cycle de transitions de A, admet une durée supérieure à d. Nous pouvons facilement établir que les conditions suivantes garantissent qu'un automate temporisé A est fortement non-zénon :

Pour chaque cycle de transitions $q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{k-1}} q_k \xrightarrow{e_k} q_1$ dans A, il existe au moins deux transitions e, e' dans ce cycle, une horloge x et un entier naturel $c \leq 1$, tels que :

1. x est remise à zéro dans e ;
2. x admet la valeur c comme une borne inférieure dans la contrainte de garde g' associée à la transition $e' : g' \wedge (x < c) = \emptyset$.

Dans un automate temporisé FNZ, les traces infinies sont nécessairement à temps-divergent. En effet, la progression du temps après l'exécution de chaque cycle de transitions est uniformément bornée par une durée minimale strictement positive (Roux et Rusu, 1996; Tripakis, 1999).

4.2.4 Quelques résultats sur les automates temporisés

Nous rappelons dans la suite quelques résultats intéressants sur le modèle automate temporisé.

- **Le problème du vide** : Ce problème consiste à vérifier si un automate temporisé accepte au moins une trace temporisée ; i.e., décider si $L(A)$ est vide, où A désigne un automate temporisé.
  Ce problème a été montré décidable et sa complexité est *PSPACE-Complet* (Alur et Dill, 1994).
- **Le problème d'atteignabilité (ou d'accessibilité) :** ce problème consiste à vérifier, pour un état donné (q, v) d'un automate temporisé, s'il existe une exécution de l'automate telle que (q, v) est accessible depuis l'état initial à travers cet exécution.
  Ce problème a été montré décidable et sa complexité est *PSPACE-Complet* (Alur et Dill, 1994).

- **Le problème de déterminisation :** Ce problème consiste à vérifier s'il existe une procédure permettant de déterminer l'automate temporisé déterministe équivalent à un automate temporisé donné.
  Ce problème a été montré indécidable (Alur et al., 1999).
- **Le problème de synthèse d'un diagnostiqueur :**
  Le problème de synthèse du diagnostiqueur à partir d'un modèle automate temporisé peut être réduit à un problème de déterminisation d'un automate temporisé comportant des actions silencieuses, dites aussi *ϵ-transitions* (Tripakis, 2002). Le modèle automate temporisé avec *ϵ-transitions* est une variante du modèle automate temporisé classique sur lequel on introduit des actions silencieuses dans l'alphabet utilisé. Ce modèle est plus expressif que le modèle classique (Bérard et al., 1998) et par conséquent, tout problème indécidable pour le modèle classique le sera également pour ce modèle, d'où découle l'indécidabilité du problème de déterminisation d'un modèle automate temporisé muni d'actions silencieuses.
  Dans (Bouyer et Chevalier, 2005), les auteurs se sont intéressés à l'étude de la décidabilité du problème de synthèse du diagnostiqueur à partir de sous-classes spécifiques d'automates temporisés, telles que les automates temporisés déterministes avec des ressources bornées et les automates ERA (Event Recording Automata) introduits dans (Alur et al., 1999). La sous-classe ERA impose que les horloges soient associées à des événements. Le problème de synthèse du diagnostiqueur a été montré décidable pour ces deux sous-classes.

4.2.5 Analyse Symbolique - Recherche d'accessibilité

Dans la mesure où le temps est continu, il est impossible d'énumérer tous les états d'un automate temporisé. En d'autres termes, l'espace d'état d'un automate temporisé est infini et non dénombrable. Pour pouvoir vérifier un modèle automate temporisé, il est nécessaire de disposer d'une représentation symbolique permettant de manipuler cet espace d'état.

Alur et Dill proposent dans (Alur et Dill, 1994) une abstraction de l'espace d'état basée sur la notion de *région*. Cette notion représente la base formelle qu'a permis de prouver la décidabilité des problèmes du vide et d'atteignabilité. Le principe de cette construction consiste à abstraire les comportements des automates temporisés en un ensemble de classes d'équivalence, appelés *régions*. En effet, l'ensemble infini des états est partitionné en des régions. Chaque région est constituée d'un ensemble d'états liés par une relation d'équivalence comportementale appelée relation de bisimulation. Cette relation est définie de la manière suivante. Deux états (l, v) et (l', v') sont équivalents, si (1) $l = l'$ et (2) $v \cong_K v'$, où K désigne la constante maximale qui apparaît dans l'automate. Deux valuations v et v' sont équivalentes, qu'on note $v \cong_K v'$, si les conditions suivantes sont vérifées :

1. pour toute horloge $x \in X$, $v(x) > K \Leftrightarrow v'(x) > K$;
2. pour toute horloge $x \in X$, si $v(x) \leq K$ alors $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ et $(\{v(x)\} = 0 \Leftrightarrow \{v'(x)\} = 0)$;
3. pour toutes les paires d'horloges (x, y), si $v(x) \leq K$ et $v(y) \leq K$, alors $(\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\})$;

où $\{\alpha\}$ désigne la partie fractionnaire d'un réel α et $\lfloor \alpha \rfloor$ désigne sa partie entière.

D'une manière intuitive, les deux premières conditions impliquent que deux valuations équivalentes satisfont exactement les mêmes contraintes de l'automate. La troisième condition exprime qu'à partir de deux configurations équivalentes, l'écoulement du temps permettra aux horloges d'atteindre de nouvelles valeurs entières dans le même ordre.

Exemple 4.2.3
Dans l'exemple suivant, nous illustrons un exemple de partition en régions d'un espace d'état.

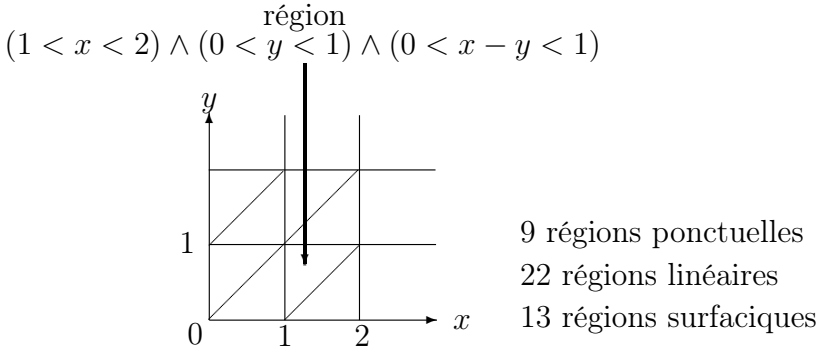

Fig. 4.2 – *Un exemple d'une région définie par deux horloges x et y*

Il est clair à travers cet exemple que la notion de région est trop fine pour être utilisée par des algorithmes d'analyse d'automates temporisés. Une autre abstraction plus efficace de l'espace d'état a été proposée dans la littérature (Alur, 1999). Cette représentation symbolique se base sur la notion de *zones*. Une zone est un ensemble de valuations défini par une contrainte de l'ensemble $\mathcal{C}(X)$.

Durant l'analyse en avant d'un automate temporisé, les objets qui seront manipulés seront des paires (q,z), appelées *états symboliques*, où q est un sommet de l'automate et z une zone. Un état symbolique peut être considéré comme une union de régions. Dans la figure 4.3, nous illustrons l'exemple d'une zone définie par deux horloges x et y. Cette zone peut être décrite par la contrainte $\langle (y \leq 4) \wedge (x \leq 5) \wedge (1 \leq x - y \leq 4) \rangle$. Nous notons que cette zone peut être représentée par des formules de contraintes différentes. Cependant, il existe une représentation unique pour chaque zone d'horloges (convexe ou non-convexe), appelée représentation canonique (Bengtsson et Yi., 2004).

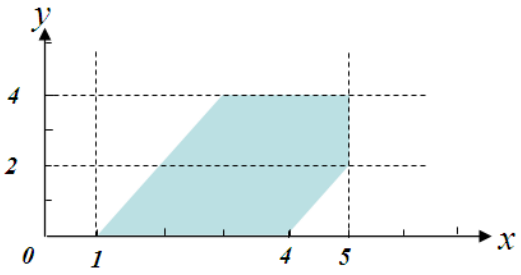

Fig. 4.3 – Un exemple d'une zone d'horloges

Nous présentons dans la dernière section de ce chapitre un aperçu sur l'implémentation des zones et les algorithmes permettant leur manipulation.

Plusieurs opérations élémentaires ont été définies sur les zones d'horloges (Bengtsson et Yi., 2004). Nous présentons dans la suite quelques unes utilisées dans notre travail.

- la remise à zéro d'un ensemble d'horloges Y dans z, définie par $z[Y \leftarrow 0] = \{v[Y \leftarrow 0] \mid v \in z\}$. Dans l'exemple de la figure 4.4, la remise à zéro de l'horloge y dans la zone d'horloges définie par la contrainte $\langle(2 \leq y \leq 4) \wedge (1 \leq x \leq 6) \wedge (-1 \leq x - y \leq 2)\rangle$ définit la zone décrite par $\langle(y = 0) \wedge (1 \leq x \leq 6)\rangle$;

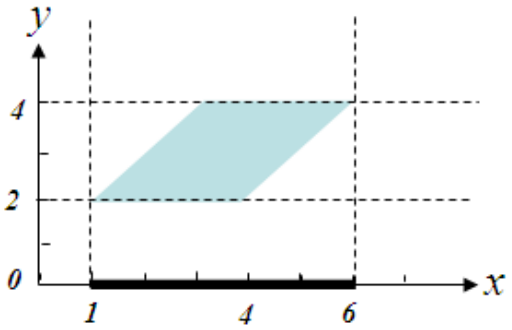

Fig. 4.4 – La remise à zéro d'une zone

- le calcul du futur d'une zone z, défini par $z \uparrow = \{v + d \mid v \in z \text{ et } d \in \mathbb{R}_+\}$. Cette opération permet d'obtenir les successeurs de temps d'un ensemble d'états défini par un état symbolique. Dans l'exemple présenté dans la figure 4.5, le futur de la zone définie par la contrainte $\langle(0 \leq y \leq 2) \wedge (1 \leq x \leq 5)\rangle$ correspond à la zone définie par $\langle(x \geq 1) \wedge (-1 \leq x - y \leq 5)\rangle$;
- l'intersection de deux zones z et z', définie par $z \wedge z' = \{v \mid v \in z \text{ et } v \in z'\}$. Dans l'exemple de la figure 4.6, l'intersection de la zone définie par $\langle(y \leq 4) \wedge (1 \leq x - y \leq 4)\rangle$ et la zone $\langle(4 \leq x \leq 7) \wedge (2 \leq y \leq 5)\rangle$ est égale à la zone définie par $\langle(2 \leq y \leq 4) \wedge (4 \leq x \leq 6) \wedge (x - y \leq 4)\rangle$.

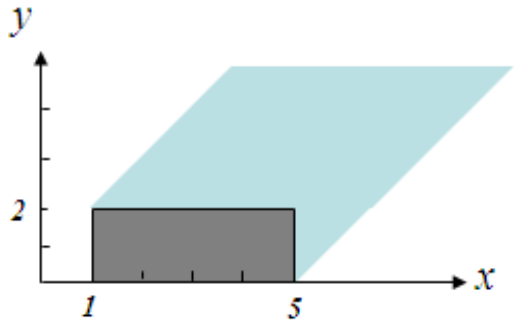

Fig. 4.5 – Le futur d'une zone

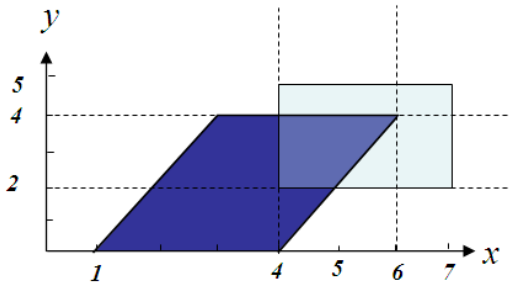

Fig. 4.6 – Une intersection de deux zones

Soient $q \xrightarrow{\sigma,g,Y} q' \in E$ une transition d'un automate temporisé et (q,z) un état symbolique. L'état symbolique (q,z) peut évoluer selon deux manières :

- *les transitions de temps :* définies par l'opérateur $Post_t((q,z)) = (q,z')$ où $z' = \{v' \mid \exists d \in \mathbb{R}_+, v' = v + d, v' \in I(q), v \in z\} = z \uparrow \wedge I(q)$.
- *les transitions d'actions :* définies par l'opérateur $Post_d((q,z), \xrightarrow{\sigma,g,Y}) = (q',z')$, où $z' = \{v' \mid (q,v) \xrightarrow{\sigma} (q',v'), v \in z\} = ((z \wedge g)[Y \leftarrow 0]) \wedge I(q')$.

On définit le successeur d'un état symbolique (q,z) sur une transition $q \xrightarrow{\sigma,g,Y} q' \in E$ par :

$$Post((q,z), \xrightarrow{\sigma,g,Y}) = Post_t \circ Post_d((q,z), \xrightarrow{\sigma,g,Y})$$

Généralement, l'étude d'un système modélisé par un automate temporisé est basée sur l'analyse de l'atteignabilité des états de l'automate. Pour savoir si un ensemble d'états d'arrivée, défini par un état symbolique (q',z') est atteignable depuis un ensemble d'état de départ (q,z), nous pouvons appliquer une analyse d'atteignabilité à partir de l'ensemble des états de départ. Cette analyse permet de calculer tous les états qui peuvent être accessibles depuis les états définis par le couple (q,z). Si l'espace calculé contient des états qui appartiennent également à l'ensemble d'états d'arrivée (q',z'), alors on peut dire que l'ensemble d'arrivée est atteignable depuis l'ensemble de départ. L'opérateur *Post*, défini ci-dessus, permet d'établir l'analyse d'atteignabilité en avant. En effet, il

suffit d'appliquer itérativement cet opérateur, depuis l'état symbolique de départ, en considérant les différentes transitions discrètes possibles. L'analyse en avant d'un automate temporisé ne se termine pas en général. Dans (Desel et al., 2004), nous pouvons trouver un exemple d'une telle analyse en avant inachevable.

Afin de forcer la terminaison de ce calcul, un opérateur d'extrapolation (ou d'approximation) des zones d'horloges est généralement utilisé. Cette opération, appelée aussi normalisation, consiste à ignorer la valeur précise d'une horloge si elle dépasse un entier K, défini comme la plus grande constante existante dans les expressions des gardes et des invariants de l'automate. En effet, il suffit de retenir que cette valeur est plus grande que K (Bengtsson et Yi., 2004).

4.3 Contexte et objectifs de notre approche de diagnostic à base de modèles temporisés

Dans ce travail, nous nous basons sur les travaux de Sampath (Sampath et al., 1995, 1996), qui propose la compilation d'un diagnostiqueur à partir d'un modèle automate à états finis du système. Notre objectif consiste à développer une nouvelle approche de diagnostic dans le cadre des modèles temporisés à temps continu (par opposition aux modèles à temps discret). Dans la figure 4.7, nous illustrons le schéma global de l'approche de diagnostic que nous allons développer.

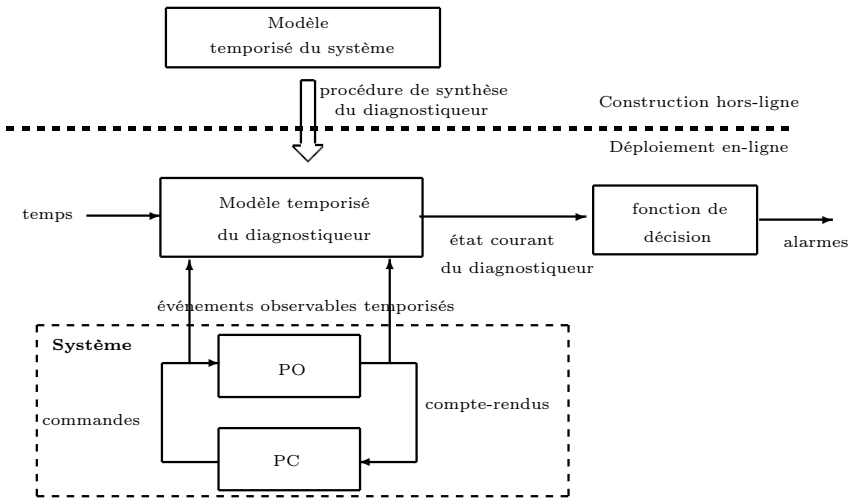

Fig. 4.7 – *Schéma global de notre approche de diagnostic à base d'automates temporisés*

Comme une hypothèse de départ, nous supposons que nous disposons d'une part, du **modèle du système** à diagnostiquer décrivant les comportements événementiel et temporel du système et d'une autre part, d'une **observation partielle** des événements de

ce système. Le modèle du système est supposé être "complet" dans le sens où il présente tous les comportements normaux et anormaux (défaillants) du système. Nous désirons **construire le diagnostiqueur** de ce système, sous la forme d'un modèle temporisé. Le diagnostiqueur observe, en-ligne, le système pour estimer son état et les défauts pouvant affecter son fonctionnement. En se basant sur cette estimation, une fonction de décision déclenche une alarme quand un défaut certain est identifié.

Le rôle du diagnostiqueur consiste à **inférer les occurrences des défauts non observables** en se basant sur les **événements observables** ainsi que les **délais** écoulés entre ces événements. Nous souhaitons maximiser l'exploitation des contraintes temporelles existantes dans le modèle du système, afin de pouvoir discriminer tout mode de défaillance potentiel.

Dans la suite, nous illustrons les différents scénarios de discrimination de défauts que doit supporter notre approche. En effet, pour permettre le diagnostic d'un SED, notre approche repose sur des méthodes de discrimination combinant l'aspect événementiel et l'aspect temporel de l'évolution d'un SED. D'abord, nous rappelons que la perturbation du comportement d'un SED qui se produit suite à l'occurrence d'un défaut, peut correspondre à une perturbation purement événementielle (apparition d'un nouvel événement ou le changement de l'ordre des événements qui suivent le défaut) et/ou temporelle (changement des dates d'occurrence des événements qui suivent le défaut). En effet, nous pouvons distinguer trois scénarios de discrimination de défauts, déterminés par la nature de la perturbation observée (temporelle ou événementielle) :

(1) **Discrimination purement événementielle** c'est sur ce principe que repose les travaux de Sampath (Sampath et al., 1995, 1996) dans le cadre du diagnostic à base de modèles logiques. En effet, le diagnostic d'un SED est établi en utilisant uniquement une séquence d'événements observables non temporisée qui suit l'occurrence d'un défaut. Un défaut peut être identifié uniquement si son occurrence affecte, d'une manière unique et discriminante, la trajectoire des événements observables succédant le défaut. A titre d'exemple, cette approche peut être appliquée pour permettre le diagnostic d'un débordement dans un bassin en phase de remplissage. En effet, un capteur de niveau d'urgence émet une alarme lors du débordement. Cet événement (l'alarme du capteur) permet de distinguer la trajectoire défaillante du système, sans ayant recours à d'autres informations comme la date d'occurrence de cet événement.

Il faut noter que cette approche ne sera plus d'une grande utilité si les trajectoires à distinguer admettent des projections observables identiques d'un point de vue logique (même ordre des événements dans les séquences observées). Dans ce cas, il faut prendre en considération les dates d'occurrence des événements observables, ce qui nous conduit à définir le deuxième scénario de discrimination.

(2) **Discrimination temporelle par le biais d'événements temporisés**

Afin d'illustrer cette méthode de discrimination, nous allons considérer le système de chauffage de liquides, défini dans l'exemple 4.2.1. Nous supposons à cette étape, que le fonctionnement de ce système peut être affecté par un défaut qui correspond à l'existence d'une fuite dans le bac. Cet événement de défaut est représenté par l'événement *fuite* dans le modèle automate temporisé de ce système, illustré dans la figure 4.8. La transition sur cet événement de défaut non observable est modélisée par une **flèche en pointillés**. De même, nous modélisons, dans le reste de ce manuscrit, les transitions sur les événements non observables par des flèches en pointillés.

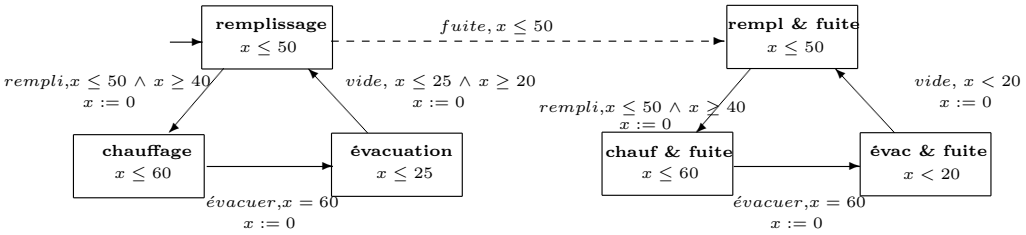

Fig. 4.8 – *Modèle du système de chauffage de liquides avec un seul défaut.*

Selon la gravité de cette fuite, le niveau du liquide dans le bac baissera d'une manière progressive. En admettant qu'il s'agit d'une petite fuite, l'occurrence de ce défaut va affecter légèrement la date d'occurrence de l'événement *rempli* qui se produit lorsque $x \in [40, 50]$ (pareil que le cas du fonctionnement normal). Pendant la phase de chauffage, le niveau du liquide va baisser d'une manière considérable dans le fonctionnement affecté par la fuite. En effet, à la fin de la phase d'évacuation, l'événement *vide*, généré par le capteur de niveau S_2, se produit à une date définie par $(x < 20)$. Par contre, cet événement se produit à une date postérieure, définie par $x \in [20, 25]$, dans le comportement normal. Ainsi, la date d'occurrence de l'événement *vide* permet la discrimination du comportement défaillant.

(3) Discrimination temporelle sur le dépassement d'un seuil d'attente

Comme dans le cas précédent, nous allons illustrer ce scénario de discrimination à travers l'introduction d'un autre cas de défaut. En effet, nous introduisons sur le modèle illustré de la figure 4.9, un cas de défaut correspondant au blocage de la vanne V_1 en position fermée. Suite à l'occurrence de ce défaut, la phase de remplissage ne va pas s'achever et par conséquent, l'événement *rempli* ne sera pas généré par le capteur de niveau S_1. Dans le modèle considéré, nous modélisons l'état défaillant qui suit l'occurrence de l'événement de défaut $blocage\_V_1$ par le sommet final **vanne bloquée**. En effet, lorsque le système évolue vers ce sommet, il ne peut plus progresser vers un autre sommet.

La discrimination de ce scénario de défaut peut être établi à travers le dépassement d'un délai seuil d'attente sans observer des événements. Ainsi, si l'événement *rempli* n'est pas observé au bout de 50 u.t., nous pouvons conclure que le système a évolué vers

le sommet **vanne bloquée**. Ainsi, le dépassement d'un seuil d'attente nous a permis d'identifier ce défaut.

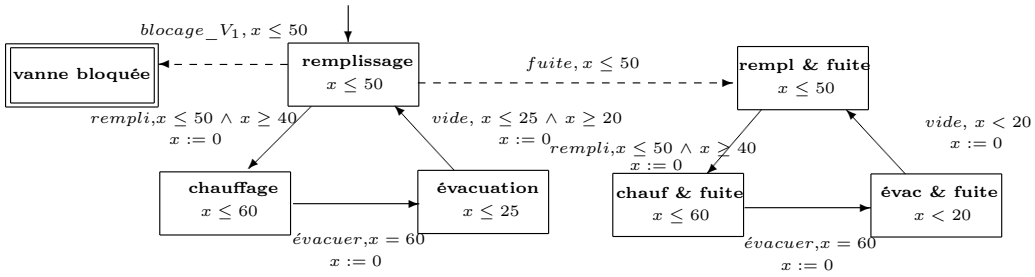

Fig. 4.9 – *Modèle du système de chauffage de liquides avec deux défauts*

Remarque 1. Dans les modèles illustrés dans les figures 4.8 et 4.9, les événements des défauts peuvent se produire uniquement pendant la phase de remplissage (à partir du sommet **remplissage**). Cependant, il est possible que ces défauts se produisent pendant la phase de chauffage ou d'évacuation. Le choix de ne pas représenter certaines transitions de défaut est considéré dans le but d'alléger la structure du modèle permettant ainsi de simplifier la compréhension de notre approche, qui reste toutefois applicable pour un modèle plus complet.

4.4 Méthode de diagnostic à base d'automates temporisés

Nous présentons dans cette section les différentes étapes de notre méthode de diagnostic à base d'automates temporisés. Dans une première étape, nous caractérisons les spécifications et les hypothèses retenues sur le modèle automate temporisé du système à diagnostiquer. Dans une seconde étape, nous illustrons l'algorithme de synthèse du diagnostiqueur qui représente le cœur de notre approche de diagnostic.

4.4.1 Modèle temporisé pour le diagnostic : spécifications et hypothèses

Dans notre travail, nous avons choisi le formalisme des automates temporisés comme outil de base pour la modélisation des systèmes à diagnostiquer. Ce choix est justifié par la généralité qu'offrent les automates temporisés, leur capacité à représenter les contraintes temporelles liées à l'occurrence des événements, leur pouvoir d'expression et les possibilités de modélisation qu'ils donnent (Alur, 1999). En plus, ces modèles bénéficient d'une grande capacité d'analyse (Bengtsson et Yi., 2004). En effet, plusieurs outils

de modélisation d'analyse ont été développés dans la littérature tel que UPPAAL (Larsen et al., 1997) et KRONOS (Yovine, 1997). Par ailleurs, comme nous avons indiqué dans le chapitre précédent, certains outils de modélisation temporisés, tel que le modèle RdP temporel, peuvent être traduits à des automates temporisés, ayant un comportement équivalent, obtenus en appliquant des algorithmes de transformation (Cassez et Roux, 2006).

Nous supposons que le système à diagnostiquer est modélisé par un automate temporisé $A = (Q, X, \Sigma, E, q_0, Q^f, I)$. Par ailleurs, nous supposons que ce modèle respecte un ensemble de spécifications, caractérisant essentiellement la modélisation des défauts. Ces spécifications sont énumérées ci-dessous :

- le modèle du système est "complet", ainsi, il permet de décrire les comportements normaux et anormaux (défaillants) ;
- l'ensemble des événements Σ est réparti en deux sous-ensembles : l'ensemble des événements observables Σ_o et l'ensemble des événements non observables Σ_{uo} ;
- l'ensemble Σ_f désigne l'ensemble des événements de défauts. Nous supposons que tous les défauts pouvant affecter le système sont non observables ; i.e., $\Sigma_f \subseteq \Sigma_{uo}$;
- nous supposons que l'ensemble des défauts Σ_f forme une partition de m sous-ensembles disjoints et non-vides de défauts (ou modes de défaillance) $\Sigma_f = F_1 \cup \cdots \cup F_m$;
- les défauts sont supposés être permanents. En effet, le système ne peut pas retourner au comportement normal après l'occurrence d'un défaut ;
- deux défauts, appartenant à deux sous-ensembles distincts, ne peuvent pas se produire dans une même exécution du système. Par conséquent, le scénario de défauts multiples, qui correspond à l'occurrence multiple et successive de défauts provenant de différents modes, n'est pas considéré dans notre approche. Une extension de notre approche de diagnostic supportant les défauts multiples peut être envisagée comme une perspective de notre travail ;
- à son état initial, nous supposons que le système est dépourvu de défauts. Ainsi, cet état fait partie de l'ensemble des états du comportement normal.

Dans la suite, nous considérons une propriété permettant de caractériser une des hypothèses à considérer sur le modèle du système. Intuitivement, cette propriété stipule que : si deux chemins dans un automate temporisé ont les mêmes séquences de transitions observables, alors ils doivent avoir des ensembles de réinitialisation d'horloges identiques

sur ces transitions.

Propriété 1. Soit A un automate temporisé et $p : q_0 \xrightarrow{\sigma_1,g_1,Y_1} q_1 \xrightarrow{\sigma_2,g_2,Y_2} q_2 \dots q_{k-1} \xrightarrow{\sigma_k,g_k,Y_k} q_k$ et $\tilde{p} : \tilde{q}_0 \xrightarrow{\tilde{\sigma}_1,\tilde{g}_1,\tilde{Y}_1} \tilde{q}_1 \xrightarrow{\tilde{\sigma}_2,\tilde{g}_2,\tilde{Y}_2} \tilde{q}_2 \dots \tilde{q}_{k'-1} \xrightarrow{\tilde{\sigma}_{k'},\tilde{g}_{k'},\tilde{Y}_{k'}} \tilde{q}_{k'}$, deux chemins dans A possédant le même nombre n de transitions sur des événements observables (ou simplement transitions observables). On note par σ_i^o (respect. $\tilde{\sigma}_i^o$) et Y_i^o (respect. $\tilde{Y}_i^o$) l'événement et l'ensemble d'horloges à réinitialiser de la $i^{ème}$ transition observable dans p (respect. dans $\tilde{p}$), $i \in \{1, \dots, n\}$. La propriété 1 est vérifiée si :

$$(\exists j \in \{1, \dots, n\}, \forall i \in \{1, \dots, j\}, \sigma_i^o = \tilde{\sigma}_i^o) \Rightarrow (\forall i \in \{1, \dots, j\}, \quad Y_i^o = \tilde{Y}_i^o)$$

Afin de mieux présenter cette propriété, nous considérons l'exemple suivant :

Exemple 4.4.1

Dans l'automate temporisé illustré dans la figure 4.10, nous supposons que $a, b, c, d, e \in \Sigma_o$ et que $u, f \in \Sigma_{uo}$.

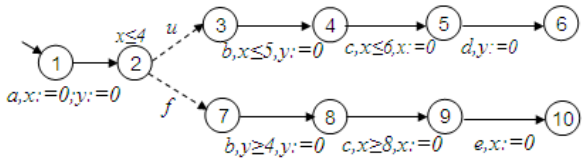

Fig. 4.10 – Exemple d'un automate temporisé vérifiant la Propriété 1

Par une simple inspection de cet automate temporisé, nous pouvons affirmer qu'il vérifie la propriété 1. En effet, les transitions sur les événements b et c, dans les chemins $1 \xrightarrow{a} 2 \xrightarrow{u} 3 \xrightarrow{b} 4 \xrightarrow{c} 5$ et $1 \xrightarrow{a} 2 \xrightarrow{f} 7 \xrightarrow{b} 8 \xrightarrow{c} 9$, admettent les mêmes ensembles de réinitialisation d'horloges. Les réinitialisations d'horloges dans les transitions sur les événements d et e peuvent être différentes.

La vérification de cette propriété est simple à effectuer sur cet exemple. Par contre, elle sera plus compliquée à réaliser pour des modèles de plus grandes tailles. Ainsi, il faut disposer d'une technique systématique permettant la vérification de cette propriété. Pour ce faire, nous proposons l'application d'un algorithme inspiré de la procédure de déterminisation des automates à états finis (Cassandras et Lafortune, 1999). Dans une première étape, nous éliminons toutes les contraintes de gardes et d'invariants définies dans l'automate. Ensuite, nous appliquons l'algorithme de déterminisation d'automates à états finis, en considérant les événements non observables comme des actions silencieuses. Cette construction permet de définir une classe d'équivalence sur l'ensemble des sommets

de l'automate, où chaque paire de sommets équivalents est accessible par deux chemins vérifiant la propriété 1. Pendant cette opération de déterminisation, si deux transitions sur le même événement observable, issues de deux sommets équivalents, admettent des ensembles de réinitialisation différents, nous arrêtons l'algorithme et nous annonçons que la propriété 1 n'est pas vérifiée. Lorsque l'algorithme se termine normalement, nous annonçons que le modèle vérifie la propriété.

La figure 4.11 présente une application de l'algorithme de vérification de la propriété 1 sur le modèle de la figure 4.10.

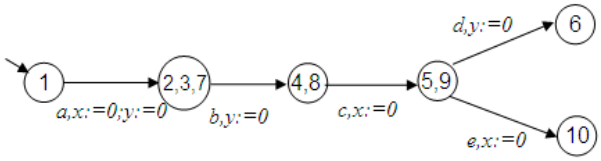

Fig. 4.11 – Une application de l'algorithme de vérification de la Propriété 1

Hypothèses :

Nous supposons que le modèle du système A satisfait les hypothèses suivantes :

$\mathcal{H}_1$) A est fortement non-zénon ;

$\mathcal{H}_2$) aucune affectation d'horloges n'est autorisée dans les transitions sur des événements non observables ;

$\mathcal{H}_3$) les affectations d'horloges sur les événements observables doivent respecter la propriété 1 ;

$\mathcal{H}_4$) les horloges $\{x_1, \ldots, x_n\}$ sont bornées par une constante entière positive K, dans chaque sommet q non-final ; i.e., $I(q) = x_1 \leq K \wedge \cdots \wedge x_n \leq K$, $q \in Q - Q^f$, et sont non bornées dans chaque sommet final ; i.e., $I(q) = vrai$, $q \in Q^f$.

Nous remarquons que les hypothèses $\mathcal{H}_2$ et $\mathcal{H}_3$ permettent de caractériser les ensembles de réinitialisation d'horloges définis dans les transitions de l'automate temporisé. Nous notons que le mise à zéro aléatoire d'horloges constitue une des principales sources d'indécidabilité du problème de déterminisation des automates temporisés (Bérard et al., 1998). En effet, les auteurs présentent dans (Alur et Dill, 1994) un exemple où la déterminisation d'un automate temporisé, comportant une réinitialisation d'horloge sur une ϵ-transition, nécessite un nombre infini d'horloges.

L'hypothèse $\mathcal{H}_1$ garantit l'existence d'une borne inférieure pour l'exécution de chaque cycle de l'automate temporisée et par conséquent, la divergence du temps de chaque exécution. Cette hypothèse est intrinsèquement vérifiée dans les systèmes réels où la progression temps diverge.

Enfin, l'hypothèse $\mathcal{H}_4$ restreint la durée de séjour dans chaque sommet non final de l'automate temporisé. Aucune restriction n'est imposée pour le séjour dans les sommets finaux. Ainsi, l'automate peut séjourner indéfiniment dans un sommet final, sans pouvoir progresser vers un autre sommet. C'est pourquoi on peut qualifier les sommets finaux par *sommets puits*. Cette hypothèse est très importante puisqu'elle nous permet de délimiter l'espace d'état du système. Ainsi les valuations des états correspondant à des sommets non-finaux sont incluses dans un hypercube (Allaham, 2008) de dimension K. En considérant cette hypothèse, l'analyse d'atteignabilité appliquée au modèle considéré se termine au bout d'un temps fini (sans besoin de normalisation de zones), en considérant que l'espace d'état définit un ensemble fini de zones d'horloges.

Remarque 2. La combinaison des hypothèses $\mathcal{H}_1$ et $\mathcal{H}_2$ conduit à l'absence de cycles de transitions sur des événements non observables. En effet, à partir d'un sommet non-final, une transition sur un événement observable se produit toujours au bout d'un délai fini.

Exemple 4.4.2

Nous pouvons constater, par une simple inspection de l'automate temporisé dans la figure 4.9, que ce dernier vérifie les spécifications de modélisation considérées. En effet, ce modèle comporte deux événements de défauts $fuite$ et $blocage\_V_1$. Ces deux défauts sont permanents (pas de transitions de retour vers le comportment normal).

Nous allons vérifier dans ce qui suit, si ce modèle satisfait les hypothèses $\mathcal{H}_{i\in\{1,\ldots,4\}}$:

1. $\mathcal{H}_1$ est **satisfaite** par le modèle. En effet, chaque transition sur un événement non observable comporte un ensemble de réinitialisation d'horloges vide ;
2. $\mathcal{H}_2$ est **satisfaite** par le modèle. En effet, l'horloge x est réinitialisée dans deux transitions appartenant aux deux cycles du modèle. En plus, nous constatons que chacun de ces deux cycles comporte une transition ayant une contrainte de garde inférieurement bornée par la valeur 40 ;
3. $\mathcal{H}_3$ est **satisfaite** par le modèle. La propriété 1 est satisfaite par ce modèle puisque les deux cycles : celui du comportement normal et celui du comportement défaillant, admettent le même ensemble d'affectation d'horloges ($x := 0$) dans toutes les transitions.
4. $\mathcal{H}_4$ **n'est pas satisfaite** par le modèle. En effet, les sommets non finaux admettent différentes conditions d'invariants. Pour pallier à ce problème, nous proposons de borner l'horloge x dans tous les sommets non-finaux par la valeur 60. Ainsi, nous obtenons le modèle illustré dans la figure 4.12, qui **vérifie** l'hypothèse $\mathcal{H}_4$.

Remarque 3. Comme dans cet exemple, nous sommes contraints parfois de changer les conditions d'invariants afin que le modèle du système respecte l'hypothèse $\mathcal{H}_4$, ce qui

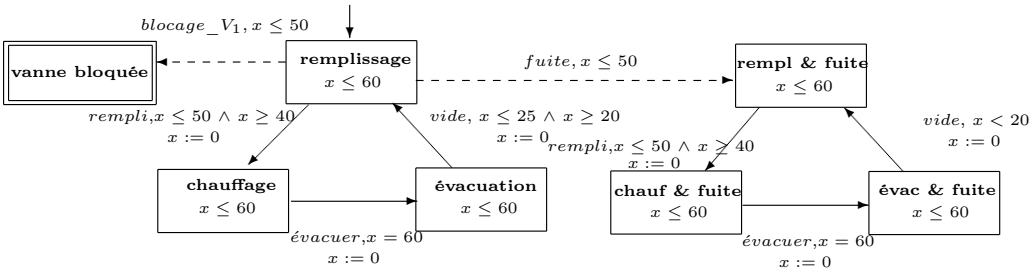

Fig. 4.12 – *Modèle du système de chauffage de liquides vérifiant toutes les hypothèses de modélisation*

conduit à un changement dans l'espace d'état accessible. Une question importante se pose suite à ces modifications : est-ce que ces changements sont sans conséquences sur l'application de notre approche de diagnostic ?

Afin de répondre à cette question, il faut préciser en premier lieu que les conditions d'invariants permettent essentiellement de garantir la progression de l'exécution d'un automate temporisé. Ainsi, il est possible de séjourner dans un sommet de l'automate tant que la valuation courante des horloges satisfait la condition d'invariant du sommet. Une fois que cette condition est violée, une des transitions de sortie doit être franchie.

Dans le cadre de notre méthode de construction du diagnostiqueur, le changement de ces contraintes d'invariants n'a pas généralement d'impact sur le modèle du diagnostiqueur obtenu, si la condition d'invariant initiale est bornée. En effet, notre algorithme de synthèse du diagnostiqueur se base essentiellement sur les contraintes associées aux gardes des transitions, pour déterminer l'ensemble des états physiquement atteignables. Toutefois, si nous désirons spécifier des contraintes d'invariants dans le but de réduire l'espace de séjour du système dans un sommet déterminé, nous pouvons décaler la contrainte d'invariant du sommet dans les gardes des transitions en aval (transitions de sorties du sommet). Si la condition d'invariant initiale n'est pas bornée, il faut revoir la conception du modèle pour avoir des conditions d'invariants bornées sur les sommets non-finaux.

4.4.2 Structure du diagnostiqueur

Nous présentons dans cette partie la structure du diagnostiqueur que nous désirons construire à partir d'un automate temporisé vérifiant les hypothèses et les spécifications énumérées ci-dessus. La structure de notre diagnostiqueur est inspirée du modèle du diagnostiqueur proposé dans les travaux de Sampath (Sampath et al., 1995, 1996).

Notre diagnostiqueur est un automate temporisé déterministe, compilé à partir du modèle du système. Cet automate est exécuté en-ligne avec le système pour permettre l'élaboration du diagnostic. Une évolution d'un sommet à un autre dans cet automate se

fait uniquement par le biais de transitions sur des événements observables. Par ailleurs, le diagnostiqueur peut évoluer vers un autre sommet suite à l'écoulement d'un délai seuil d'attente à travers des transitions urgentes (Barbuti et Tesei, 2004).

Les sommets du diagnostiqueur correspondent à des **macro-états** (Ghazel, 2005), pouvant contenir plusieurs ensembles d'états du système enrichis par des informations relatives aux défauts. En effet, après l'occurrence de chaque événement observable, le diagnostiqueur évolue vers le prochain sommet qui fournit une estimation de l'état du système, à l'instant du franchissement de cet événement. Par ailleurs, à chacun de ces états est associé une étiquette de diagnostic permettant d'indiquer si un défaut s'est produit dans l'exécution menant vers cet état.

Dans la suite, nous présentons une définition formelle de l'automate temporisé du diagnostiqueur.

Nous notons par $\Phi = \{\mathcal{N}, \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m\}$, un ensemble d'étiquettes, dites *étiquettes de diagnostic*. Ces étiquettes permettent de déterminer le mode de défauts affectant un ensemble d'états du système. En effet, un ensemble d'états du système est qualifié de *F_i-défaillant*, s'il est associé à une étiquette $\mathcal{F}_i$. Autrement dit, un défaut de l'ensemble F_i s'est produit avant d'atteindre un état de cet ensemble. De même, un ensemble d'états est qualifié de *normal*, s'il est associé à l'étiquette $\mathcal{N}$. Dans ce dernier cas, aucun défaut ne s'est produit avant d'atteindre un état de cet ensemble.

Définition 25. Un diagnostiqueur est un automate temporisé déterministe $D = (Q^d, X, \Sigma_o, E^d, q_0^d, Q_f^d, I)$, où :

- $Q^d \subset 2^{Q\times\mathcal{F}} \times \mathcal{C}(X)$ est un ensemble de sommets du diagnostiqueur. Chaque sommet du diagnostiqueur est défini par un ensemble $q^d = (\{(q_i, \phi_i), i \in \{1, \ldots, k\}\}, z)$, où z désigne une zone d'horloges et (q_i, ϕ_i), $i \in \{1, \ldots, k\}$ correspond à un ensemble de couples "sommet/étiquette de diagnostic".
- E^d est un ensemble fini de transitions. Une transition entre deux sommets du diagnostiqueur q_d et q'_d est définie par un quintuplet $(q_d, \sigma, g, Y, q'_d)$, où σ désigne un événement observable ; i.e., $\sigma \in \Sigma_o$, et g est une contrainte de garde de l'ensemble $\mathcal{C}(X)$.
- q_0^d désigne le sommet initial du diagnostiqueur. Ce sommet est défini comme suit : $q_0^d = (\{(q_0, \mathcal{N})\}, z_0)$, où q_0 désigne le sommet initial du modèle du système et z_0 désigne la zone d'horloge initiale. Cette zone ponctuelle associe la valeur 0 pour toutes les horloges de l'ensemble X ; i.e., $x_1 = x_2 = \cdots = x_n = 0$. L'étiquette $\mathcal{N}$ indique que cet état initial fait partie du comportement normal du système.
- Q_f^d désigne l'ensemble des sommets finaux,
- I est la fonction qui associe une contrainte d'invariant à chaque sommet.

Remarque 3. Nous soulignons que, contrairement à la définition originale des automates temporisés introduite dans (Alur et Dill, 1994), les conditions des gardes des transitions définies dans l'automate temporisé du diagnostiqueur contiennent des contraintes diago-

nales de la forme $x - y \sim c$. L'implémentation du diagnostiqueur reste toujours possible en considérant ces contraintes.

Définition 26. Soit $q^d = (\{(q_1, \phi_1), \ldots, (q_k, \phi_k)\}, z)$ un sommet d'un diagnostiqueur. Nous définissons l'opérateur Dis, qui renvoie la partie discrète d'un sommet du diagnostiqueur ; i.e. $Dis(q^d) = \{(q_1, \phi_1), \ldots, (q_k, \phi_k)\}$. De même, l'opérateur Z renvoie la zone d'horloges associée à un sommet du diagnostiqueur ; i.e., $Z(q^d) = z$.

Définition 27. Un sommet du diagnostiqueur $q^d = (\{(q_1, \phi_1) \ldots (q_k, \phi_k)\}, z)$ est dit :

1. F_i-*certain*, si $\phi_p = \mathcal{F}_i$, $\forall p \in \{1, \ldots, k\}$.

2. F_i-*incertain*, s'il existe $(q, \phi), (\tilde{q}, \tilde{\phi}) \in Dis(q^d)$, où $\phi = \mathcal{F}_i$ et $\tilde{\phi} \neq \mathcal{F}_i$.

Intuitivement, un sommet du diagnostiqueur est F_i-certain lorsque tous les états estimés dans ce sommet sont F_i-défaillants. Lorsque la fonction de diagnostic détecte que le sommet courant du diagnostiqueur est F_i-certain, elle génère une alarme annonçant l'occurrence d'un défaut de l'ensemble F_i. Par contre, si le sommet courant est F_i-incertain, aucune décision ne pourra être prise par cette fonction. En effet, les états estimés comportent à la fois des états F_i-défaillants et d'autres états associés à un autre mode fonctionnement (le mode de fonctionnement normal ou un mode de défaillance F_j, $j \neq i$) ce qui implique une ambiguïté dans la prise de décision.

Dans la suite, nous présentons un exemple qui illustre le diagnostiqueur du modèle du système de chauffage de liquides, présenté dans la figure 4.12. Nous détaillons ultérieurement l'algorithme de synthèse de ce diagnostiqueur.

Exemple 4.4.3
L'automate temporisé de la figure 4.13 représente le diagnostiqueur du système de chauffage de liquides, obtenu à partir du modèle de la figure 4.12. Chaque sommet de ce diagnostiqueur comporte un ensemble de paires (sommet,étiquette), associé à une contrainte sur l'horloge x qui définie la valuation de cette horloge après le franchissement de la transition d'entrée du sommet.

Afin de construire ce diagnostiqueur, nous avons considéré la partition de défauts suivante : $\Sigma_f = F_1 \cup F_2$, où $F_1 = \{fuite\}$ correspond à l'occurrence d'une fuite et $F_2 = \{blocage\_V_1\}$ correspond au blocage de la vanne V_1. Nous allons considérer deux cas d'application de ce diagnostiqueur. Nous supposons, dans un premier cas, que le système effectue une exécution sur la trace : ($fuite$,10)($rempli$,37.4)($évacuer$,60)($vide$,13). En observant la projection observable de cette trace : ($rempli$,47.4)($évacuer$,60)($vide$,13), le diagnostiqueur évolue vers le sommet ($\{$(**rempl. & fuite**,$\mathcal{F}_1$)$\}$, $x = 0$). La fonction de décision constate que le sommet est F_1-certain et génère, par conséquence, une

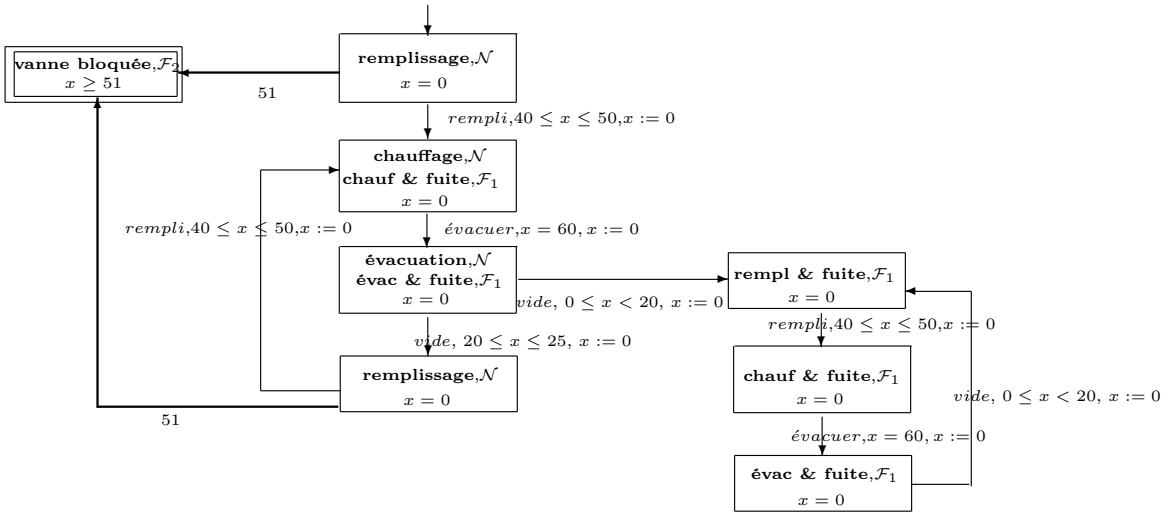

Fig. 4.13 – *Exemple du diagnostiqueur d'un système de chauffage de liquides*

alarme indiquant l'occurrence d'un défaut de l'ensemble F_1. Nous remarquons pour pour n'importe quelle suite d'événements observés, le diagnostiqueur va évoluer dans des sommets F_1-certain. Ceci est attendu puisque les défauts considérés dans le modèle du système sont permanents.

Nous supposons, dans un deuxième cas, que le système effectue une exécution sur la trace : ($rempli$,43.4)($évacuer$,60)($vide$,23.4)($blocage\_V_1$,4). Le diagnostiqueur accepte la trace ($rempli$,43.4)($évacuer$,60)($vide$,23.4), puis il ne reçoit plus d'événements. Lorsque 51 u.t. s'écoulent après l'occurrence de l'événement $vide$, une transition urgente est franchie vers le sommet ($\{$(**vanne bloquée**,$\mathcal{F}_2$)$\}, x \geq 51$) dans le diagnostiqueur. Ensuite, la fonction de décision constate que le sommet courant est F_2-certain puis génère une alarme indiquant l'occurrence d'un défaut de l'ensemble F_2.

Remarque 4. Dans cet exemple, nous avons utilisé dans l'automate temporisé du diagnostiqueur la notion de transitions urgentes. La sémantique des transitions urgentes n'a pas été spécifiée dans la définition présentée du modèle automate temporisé. Une transition urgente est prioritaire. Elle est franchie dès que sa condition de garde est satisfaite, sans besoin d'être synchronisée avec l'occurrence d'un événement (Barbuti et Tesei, 2004). Dans notre modèle, une transition urgente est étiquetée par un entier naturel qui détermine le temps de séjour maximal dans son sommet de départ, avant qu'elle soit franchie. L'implémentation d'une telle transition revient, dans une première étape, à définir une horloge, qu'on va désigner par y, permettant de mesurer le temps écoulé dans chaque sommet du diagnostiqueur. En effet, cette horloge est remise à zéro après le franchissement de chaque transition dans le diagnostiqueur. Dans une deuxième étape, nous associons à chaque transition urgente, étiquetée par un entier τ, une garde de la forme $y = \tau$. Par ailleurs, nous associons au sommet de départ de cette transition

l'invariant $y \leq \tau$ et un événement non-observable arbitraire (pour être conforme à la syntaxe définie). Ainsi, une fois que la valuation de l'horloge y atteint la valeur τ, cette condition d'invariant force le franchissement de la transition urgente.

4.4.3 Synthèse du diagnostiqueur

Nous présentons dans cette partie notre méthode de synthèse du diagnostiqueur. Nous commençons par définir un ensemble de fonctions nécessaires pour l'élaboration de cette méthode. Puis, nous exposons l'algorithme de synthèse du diagnostiqueur.

4.4.3.1 Pré-requis

- Nous définissons l'opérateur de propagation des étiquettes de diagnostic $\otimes : \Phi \times \Sigma \rightarrow \Phi$. Cet opérateur permet de déterminer l'étiquette de diagnostic associée à un état donné suite à l'occurrence d'un événement $\sigma \in \Sigma$.

$$\phi \otimes \sigma = \begin{cases} \mathcal{F}_i, & \text{if} \quad \phi = \mathcal{F}_i\,; \\ \mathcal{F}_i, & \text{if} \quad \phi = \mathcal{N} \text{ and } \sigma \in F_i\,; \\ \mathcal{N}, & \text{if} \quad \phi = \mathcal{N} \text{ and } \sigma \notin \Sigma_f. \end{cases}$$

  Nous pouvons considérer l'extension de cet opérateur pour propager une étiquette sur une séquence d'événements. Dans ce cas, il suffit d'appliquer successivement cet opérateur sur chaque événement de cette séquence, tout en propageant le résultat obtenu à chaque étape.
- Nous définissons la fonction d'accessibilité non-observable $\mathcal{UR}$ qui permet de déterminer l'ensemble d'états accessibles depuis un ensemble d'états de départ, en franchissant toutes les séquences possibles de transitions sur des événements non observables. Les étiquettes de diagnostic associées aux états de départ sont propagées durant ce calcul d'accessibilité, en utilisant l'opérateur $\otimes$.

  $\mathcal{UR} : 2^{Q \times \mathcal{C}(X) \times \Phi} \rightarrow 2^{Q \times \mathcal{C}(X) \times \Phi}$

  $[(q_i, z_i), \phi_i] \mapsto \{[(q_j, z_j), \phi_j] \mid \exists s \in (\Sigma_{uo} \times \mathbb{R}_+)^*, (q_i, z_i) \overset{s}{\rightsquigarrow} (q_j, z_j) \ et \ \phi_j = \phi_i \otimes \| s \|\}$

  Dans cette définition de la fonction $\mathcal{UR}$, l'ensemble des états de départ est donné sous la forme d'un ensemble de triplets $\{[(q_1, z_1), \phi_1], \ldots, [(q_k, z_k), \phi_k]\}$. Chaque triplet $[(q_i, z_i), \phi_i]$ représente un état symbolique (q_i, z_i) associé avec une étiquette de diagnostic ϕ_i.

  Dans Algorithme 2, nous illustrons formellement la fonction d'accessibilité non-observable $\mathcal{UR}$. Cet algorithme utilise une approche symbolique, basée sur une analyse d'atteignabilité en avant. En effet, il détermine itérativement les successeurs des états symboliques de départ, sur les possibles transitions non observables. La

terminaison de cet algorithme est garantie grâce aux hypothèses considérées sur le modèle qui assurent l'absence de cycles de transitions non observables (voir Remarque 2). Ainsi, l'algorithme se termine au bout d'un nombre fini d'itérations.

Algorithm 2 Fonction d'accessibilité non-observable $\mathcal{UR}$

Entrées : $\{[(q_i, z_i), \phi_i], i \in \{1, \ldots, k\}\}$
Initialiser $\acute{E}TATS\_ACCESSIBLES$ et $A\_TRAITER$ par $\{[(q_i, Post_t(z_i)), \phi_i],\ i \in \{1, \ldots, k\}\}$.
tantque $A\_TRAITER \neq \{\}$ **faire**
Retirer l'élément $[(q, z), \phi]$ de $A\_TRAITER$.
pour tout $q \xrightarrow{\sigma_u, g} \tilde{q} \in E, \sigma_u \in \Sigma_{uo}$ **faire**
Calculer $(\tilde{q}, \tilde{z}) := Post((q, z), \xrightarrow{\sigma_u, g})$ et $\tilde{\phi} := \phi \otimes \sigma_u$.
si $[(\tilde{q}, \tilde{z}), \tilde{\phi}] \notin \acute{E}TATS\_ACCESSIBLES$ et $\tilde{z} \neq \emptyset$ **alors**
Ajouter $[(\tilde{q}, \tilde{z}), \tilde{\phi}]$ à $A\_TRAITER$, $\acute{E}TATS\_ACCESSIBLES$.
finsi
fin pour
fin tantque
retourner $\acute{E}TATS\_ACCESSIBLES$

4.4.3.2 Architecture de l'algorithme de synthèse du diagnostiqueur

L'algorithme de synthèse du diagnostiqueur se base sur deux notions importantes, illustrées dans la figure 4.14, qui sont : les *états d'entrée* et les *états d'ombre* (Ghazel, 2005).

- Les états d'entrée correspondent aux états accessibles suite à l'occurrence d'un événement observable. Chaque sommet du diagnostiqueur est constitué par un ensemble d'états d'entrée. Ces états sont décrits par un ensemble d'états symboliques associés à des étiquettes de diagnostic.
- Les sommets d'ombre correspondent à l'ensemble d'états accessibles depuis un ensemble d'états d'entrée, suite à l'occurrence d'événements non observable et/ou l'écoulement du temps. En effet, ces états sont obtenus en appliquent l'opérateur d'accessibilité non observable $\mathcal{UR}$, sur les états symboliques d'un sommet du diagnostiqueur.

Intuitivement, la construction du diagnostiqueur repose sur une déterminisation des transitions sur les événements observables, issues des états d'ombres. Pour ce faire, l'algorithme de synthèse du diagnostiqueur procède comme suit :

- **Première étape :** déterminer l'ensemble des états d'ombre à partir des états d'entrée ;
- **Deuxième étape :** déterminer, pour chaque événement observable, le sous-

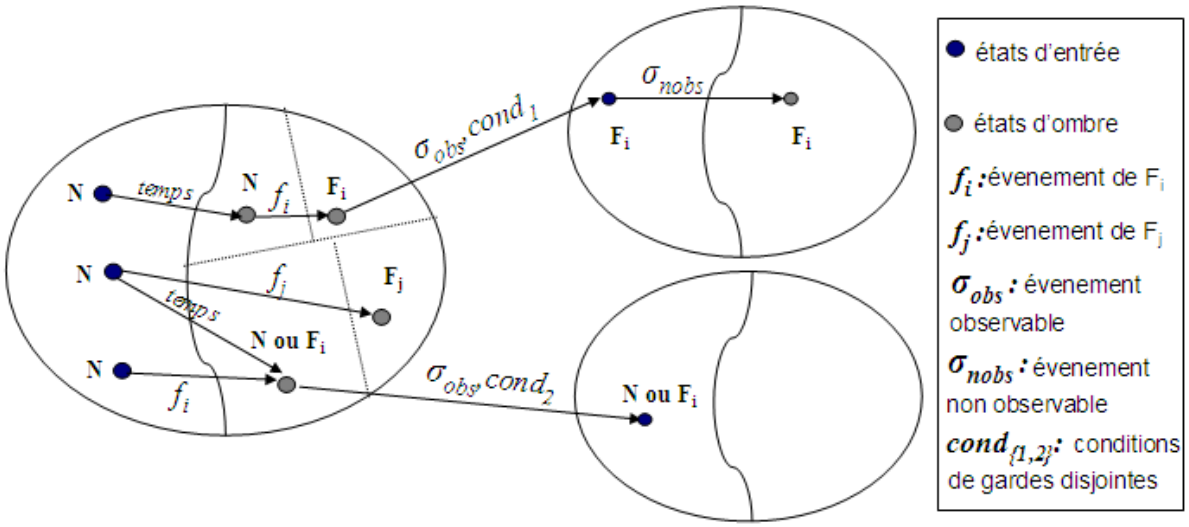

Fig. 4.14 – États d'entrée et États d'ombre

ensemble d'états d'ombre permettant de franchir une transition sur σ. Ce sous-ensemble sera appelé les *états franchissables* ;

- **Troisième étape :** estimer, pour chaque valuation d'horloges associée à l'occurrence d'un événement observable, l'ensemble des états du système. Cette étape s'appuie sur une partition de l'ensemble des états franchissables. Deux états franchissables appartenant à deux ensembles différents de cette partition ne doivent pas avoir la même valuation d'horloge. En effet, la valuation associée à l'occurrence d'un événement observable nous permet de distinguer l'ensemble des états franchissables possibles. Ces derniers doivent appartenir au même ensemble de la partition.
- **Dernière étape :** Créer, pour chaque sous-ensemble d'états de la partition des états franchissables, un nouveau sommet du diagnostiqueur. Les états d'entrée constituant ce sommet du diagnostiqueur correspondent aux successeurs discrets des états franchissables considérés.

Nous considérons dans la suite un exemple illustrant la construction d'un sommet du diagnostiqueur à partir d'une partie d'un automate temporisé. Cet exemple est décrit dans la figure 4.15.

Dans cet exemple, l'ensemble des états d'ombre de l'état initial $(q_0, x = y = 0, \mathcal{N})$, correspond aux états accessibles suite à l'écoulement du temps et/ou l'occurrence de l'événement de défaut f. En considérant $K = 5$ (borne supérieure des horloges), l'ensemble d'états d'ombre accessibles est égale à $\{(q_0, \langle x = y \wedge x \leq 5\rangle, \mathcal{N})(q_2, \langle x = y \wedge x \leq 5\rangle, \mathcal{F})\}$. L'étiquette $\mathcal{F}$ indique ici l'occurrence du défaut f.

L'ensemble des états franchissables sur l'événement a à partir des états d'ombre est égale à $\{(q_0, \langle x = y \wedge x \in [1, 3]\rangle, \mathcal{N})(q_2, \langle x = y \wedge x \in [2, 4]\rangle, \mathcal{F})\}$. Ces états sont

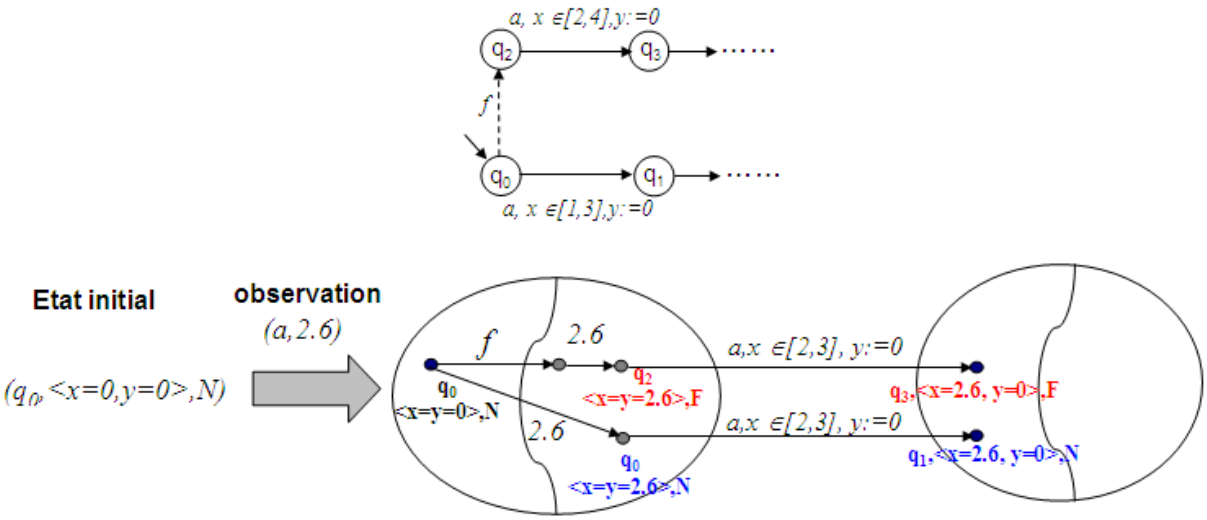

Fig. 4.15 – Principe de construction d'un sommet du diagnostiqueur

correspondent aux états d'ombres vérifiant les gardes des transitions sur l'événement a.

L'étape suivante consiste à créer une partition sur l'ensemble des états franchissables. Selon la valuation d'horloges v à l'instant de l'observation de l'événement a, nous pouvons distinguer trois cas possibles :

- si $v(x) \in [1, 2[$, l'état estimé du système correspond à $\{(q_0, v, \mathcal{N})\}$;
- si $v(x) \in [2, 3]$, l'état estimé du système correspond à $\{(q_0, v, \mathcal{N}), (q_2, v, \mathcal{F})$. Cette situation correspond au cas illustré dans notre exemple. En effet, l'événement a est observé après 2.6 u.t., les états estimés à cet instant sont : $\{(q_0, \langle x = y = 2.6\rangle, \mathcal{N}), (q_2, x = y = 2.6, \mathcal{F})$;
- si $v(x) \in]3, 4]$, l'état estimé du système correspond à $\{(q_2, v, \mathcal{F})\}$.

A partir de cette partition sur l'ensemble des états franchissables, nous pouvons construire trois sommets successeurs. Dans la figure 4.15, un seul sommet successeur est illustré. Ce sommet est obtenu en calculant les successeurs discrets de l'ensemble d'états $\{(q_0, \langle y = x \in [2, 3]\rangle, \mathcal{N}), (q_2, \langle y = x \in [2, 3]\rangle, \mathcal{F})$. Ainsi, nous obtenons le sommet $(\{(q_1, \mathcal{N}), (q_3, \mathcal{F})\}, \langle x \in [2, 3] \wedge y = 0\rangle)$.

Dans la suite, nous exposons l'ossature de l'algorithme de synthèse du diagnostiqueur d'une manière plus détaillée. Dans ce pseudo-code, nous désignons par q^d le sommet en cours d'élaboration (recherche de ses sommets successeurs) et *SOMMETS_A_TRAITER* l'ensemble des sommets qui restent à traiter. Cet ensemble est une structure de type file d'attente (FIFO). Par ailleurs, *SOMMETS_TRAITÉS* désigne l'ensemble des sommets du diagnostiqueur qui ont déjà été traités. Nous rappelons que le sommet initial $q_0^d = \{\{(q_0, \mathcal{N})\}, z_0\}$ du diagnostiqueur est constitué par le sommet initial

q_0, la zone d'horloge z_0 définie par $x_1 = x_2 = \cdots = x_n = 0$ et l'étiquette $\mathcal{N}$ indiquant que le système admet un comportement normal à son état initial. L'automate temporisé du diagnostiqueur peut être obtenu en appliquant les étapes du pseudo-code suivant :

1. initialisations :
   {
   (a) créer le sommet initial $q_0^d \leftarrow \{\{(q_0, \mathcal{N})\}, z_0\}$,
   (b) ajouter q_0^d à *SOMMETS_A_TRAITER*
   (c) *SOMMETS_TRAITÉS* $\leftarrow$ {},
   }

2. tant que (*SOMMETS_A_TRAITER*$\neq$ {})
   {
   (a) retirer un élément q^d de *SOMMETS_A_TRAITER* et l'ajouter à *SOMMETS_TRAITÉS*,
   (b) déterminer *ÉTATS_OMBRE*, l'ensemble des états d'ombre issues de q^d.
   (c) s'il existe des états finaux parmi les états d'ombre :
      i. déterminer l'entier τ qui correspond au délai minimal, après lequel aucun événement observable ne pourra être observé ; i.e., le système sera certainement dans un sommet final :

      $$\tau = min\{d \in \mathbb{N} | \forall (q, v, \phi) \in \acute{E}TATS\_OMBRE, v(y) \geq d \Rightarrow q \in Q_f\}$$

      où l'horloge y mesure le temps écoulé entre deux événements observables successifs (autrement dit, le temps de séjour dans chaque sommet du diagnostiqueur).
      ii. grouper ces états finaux dans un sommet final du diagnostiquer q_f^d, puis créer une transition urgente vers ce sommet étiquetée par le délai τ
   (d) répéter pour chaque événement observable σ.
      {
      i. déterminer à partir des états d'ombre, le sous-ensemble d'états franchissables sur σ,
      ÉTATS_FRANCHISSABLES$^\sigma$ = $\{(q, v, \phi) \in$ *ÉTATS_OMBRE* tel que il existe une transition $(q, v) \xrightarrow{\sigma} (q', v')\}$.
      ii. créer une partition sur l'ensemble *ÉTATS_FRANCHISSABLES*$^\sigma$, telle que : **(1)** un état appartient à un seul sous-ensemble de la partition et **(2)** deux états ayant la même valuation d'horloge doivent appartenir au même sous-ensemble de la partition (voir l'exemple de la partition illustrée dans figure 4.16).

iii. répéter pour chaque zone d'horloges η décrivant les valuations d'un sous-ensemble d'états franchissables de la partition :
- créer un sommet du diagnostiqueur $\tilde{q}^d$, contenant les successeurs discrets sur σ du sous-ensemble considéré de la partition d'états franchissables.
- créer une transition de q^d à $\tilde{q}^d$ ayant comme une contrainte de garde le prédicat η.
- ajouter $\tilde{q}^d$ à *SOMMETS_A_TRAITER* s'il n'a pas été encore visité ; i.e., $\tilde{q}^d \notin$ *SOMMETS_TRAITÉS*.

}

}

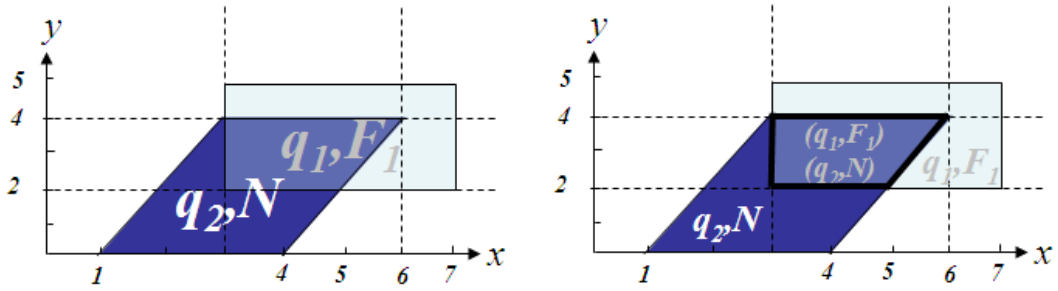

Fig. 4.16 – Création d'une partition de trois ensemble d'états franchissables.

Afin de mieux illustrer les différentes étapes de ce pseudo-code, nous allons construire dans l'exemple suivant, le diagnostiqueur correspondant à l'automate temporisé de la figure 4.17.

Exemple 4.4.4

Dans la figure 4.17, nous présentons une partie d'un automate temporisé dont on désire construire le diagnostiqueur. Il est clair que cette partie vérifie les hypothèses nécessaires pour l'application de l'algorithme de synthèse du diagnostiqueur. Les événements a et b sont observables tandis que f constitue le seul événement de défaut. Ainsi, nous aurons un seul mode de défauts, $F_1 = \{f\}$. Par ailleurs, les sommets de ce modèle sont non-finaux, auxquels nous associons la condition d'invariant $\langle x \leq 20 \wedge y \leq 20 \rangle$, où 20 est une valeur arbitrairement choisie.

La construction du diagnostiqueur se déroule comme suit :
- créer du sommet initial $(\{(1, \mathcal{N})\}, \langle x = y = 0 \rangle)$,
- à partir du sommet initial, $[(1, \langle x = y = 0 \rangle), \mathcal{N}]$, appliquer la fonction d'accessibilité non-observable afin de déterminer les états d'ombre. On

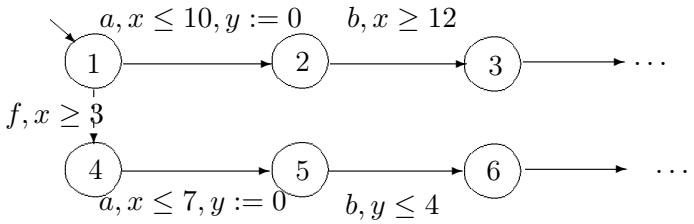

Fig. 4.17 – Exemple de construction d'un diagnostiqueur : modèle initial.

obtient *ÉTATS_ OMBRE*=$\{[(1, \langle x = y \wedge x \leq 20\rangle), \mathcal{N}], [(4, \langle x = y \wedge 3 \leq x \leq 20\rangle), \mathcal{F}_1]\}$,

- calculer les états franchissables pour les transitions sur l'événement a : $\acute{E}TATS\_FRANCHISSABLES^a = \{[(1, \langle x = y \wedge x \leq 10\rangle), \mathcal{N}], [(4, \langle x = y \wedge 3 \leq x \leq 7\rangle), \mathcal{F}_1]\}$
- déterminiser les valuations d'horloges correspondant aux états de $\acute{E}TAT\_FRANCHISSABLES^a$ afin d'avoir des zones d'horloges disjointes. On obtient les états symboliques suivants :
  - $[(1, \langle x = y \wedge x \leq 10 \wedge \neg(3 \leq x \leq 7)\rangle), \mathcal{N}]$;
  - $[(1, \langle x = y \wedge 3 \leq x \leq 7\rangle), \mathcal{N}]$;
  - $[(4, \langle x = y \wedge 3 \leq x \leq 7\rangle), \mathcal{F}_1]$;
- à partir de la partition de l'ensemble d'états franchissables, construire deux nouveaux sommets du diagnostiqueur :
  - $q_1^d = (\{(2, \mathcal{N})(5, \mathcal{F}_1)\}, \langle y = 0 \wedge 3 \leq x \leq 7\rangle)$;
  - $q_2^d = (\{(2, \mathcal{N})\}, \langle y = 0 \wedge x \leq 10 \wedge \neg(3 \leq x \leq 7)\rangle)$.
- à partir du sommet q_1^d, appliquer la fonction d'accessibilité non-observable afin de déterminer les états d'ombre. On obtient *ÉTATS_ OMBRE*=$\{[(2, \langle 3 \leq x - y \leq 7 \wedge 3 \leq x \leq 20\rangle), \mathcal{N}], [(5, \langle 3 \leq x - y \leq 7 \wedge 3 \leq x \leq 20\rangle), \mathcal{F}_1]\}$,
- calculer les états franchissables pour les transitions sur l'événement b : $\acute{E}TATS\_FRANCHISSABLES^b = \{[(2, \langle 3 \leq x - y \leq 7 \wedge 12 \leq x \leq 20\rangle), \mathcal{N}], [(5, \langle 3 \leq x - y \leq 7 \wedge x \geq 3 \wedge y \leq 4\rangle), \mathcal{F}_1]\}$ (Voir figure 4.18) ;
- les zones d'horloges correspondant aux états de $\acute{E}TATS\_FRANCHISSABLES^b$ sont déjà disjointes : pas besoin d'élaborer une partition sur cet ensemble.
- construire les sommets successeurs de q_1^d, on obtient :
  - $q_3^d = (\{(6, \mathcal{F}_1)\}, \langle 3 \leq x - y \leq 7 \wedge 0 \leq y \leq 4\rangle)$;
  - $q_4^d = (\{(3, \mathcal{N})\}, \langle 3 \leq x - y \leq 7 \wedge 12 \leq x \leq 20\rangle)$.

Enfin, nous illustrons le diagnostiqueur résultant de cette construction dans la figure 4.19. La table 4.1 définie la signification de chaque symbole utilisé dans l'automate temporisé du diagnostiqueur.

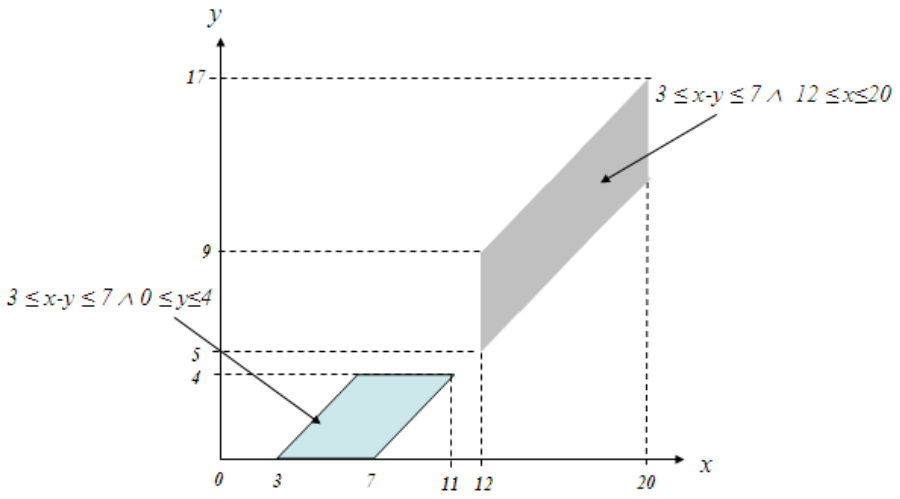

Fig. 4.18 – Exemple de construction d'un diagnostiqueur : zones disjointes des états franchissables.

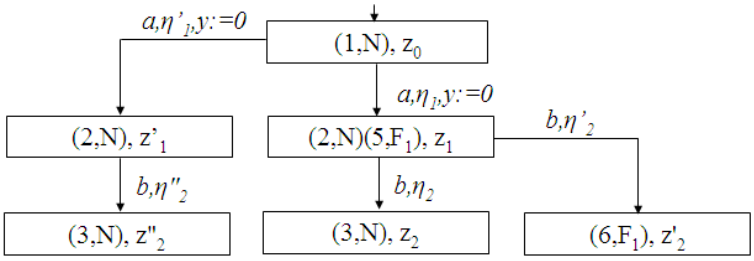

Fig. 4.19 – Exemple de construction d'un diagnostiqueur : modèle du diagnostiqueur.

La procédure formelle de construction du diagnostiqueur est illustrée dans Algorithme 3. Dans cet algorithme, l'ensemble *SOMMETS_ TRAITÉS* est désigné par l'ensemble des sommets du diagnostiqueur Q_d. Nous notons que l'invariant de chaque sommet non-final du diagnostiqueur est égal à $x_1 \leq K \wedge \cdots \wedge x_n \leq K$. En effet, l'hypothèse $\mathcal{H}_2$ empêche la réinitialisation des horloges dans les transitions atteignant les états d'ombre d'un sommet du diagnostiqueur. Nous notons que dans la ligne 9 de cet algorithme, la valeur τ est déterminée à partir d'un ensemble de valeurs $\{1, \ldots K+1\}$. Ceci est dû au fait que la valuation de l'horloge y, qui mesure le temps écoulé après l'occurrence de chaque événement observable, ne peut pas dépasser la valeur K dans les états d'ombres associés à des sommets non-finaux. Ainsi, seuls les états d'ombre associés à des sommets finaux peuvent admettre une valuation de y supérieure à K, c'est pourquoi la valeur $K+1$ correspond la valeur maximale que peut prendre τ. Par ailleurs, nous remarquons que, durant la création d'un sommet successeur du diagnostiqueur (la ligne 22), le même ensemble de réinitialisation d'horloges est considéré pour le calcul des successeurs des états franchissables. Ceci est possible grâce à l'hypothèse $\mathcal{H}_3$. En effet, les transitions

Symbole	contrainte d'horloges
z_0	$\langle x = y = 0 \rangle$
z_1	$\langle y = 0 \wedge 3 \leq x \leq 7 \rangle$
z'_1	$\langle y = 0 \wedge x \leq 10 \wedge \neg(3 \leq x \leq 7) \rangle$
z_2	$\langle 3 \leq x - y \leq 7 \wedge 12 \leq x \leq 20 \rangle$
z'_2	$\langle 3 \leq x - y \leq 7 \wedge 0 \leq y \leq 4 \rangle$
z''_2	$\langle (0 \leq x - y < 3 \wedge 12 \leq x \leq 20) \vee (7 \leq x - y \leq 10 \wedge 12 \leq x \leq 20) \rangle$
η_1	$\langle x = y \wedge 3 \leq x \leq 7 \rangle$
η'_1	$\langle x = y \wedge x \leq 10 \wedge \neg(3 \leq x \leq 7) \rangle$
η_2	$\langle 3 \leq x - y \leq 7 \wedge 12 \leq x \leq 20 \rangle$
η'_2	$\langle 3 \leq x - y \leq 7 \wedge 0 \leq y \leq 4 \rangle$
η''_2	$\langle (0 \leq x - y < 3 \wedge 12 \leq x \leq 20) \vee (7 \leq x - y \leq 10 \wedge 12 \leq x \leq 20) \rangle$

Tab. 4.1 – description des symboles utilisés dans modèle du diagnostiqueur

sur les états franchissables admettent les mêmes ensembles de réinitialisation d'horloges puisque le modèle du système vérifie la propriété 1.

4.5 Diagnosticabilité

Nous avons présenté dans la première partie de notre démarche de diagnostic les concepts liés à la synthèse et l'utilisation du diagnostiqueur. Cependant, une question importante s'impose par rapport au déploiement du diagnostiqueur : est-il capable d'identifier l'occurrence de tout défaut affectant le système au bout d'un délai fini ?

Afin de répondre à cette question, nous allons étudier la notion de diagnosticabilité des systèmes temporisés. Nous allons établir qu'un diagnostiqueur conçu à partir d'un modèle automate temporisé non diagnosticable peut aboutir à des situations ambiguës, où il ne réussit pas à identifier l'occurrence d'un défaut. Dans un tel cas, le comportement observable du système fait évoluer l'automate du diagnostiqueur vers un ensemble de sommets F_i-incertain où aucune décision sur l'occurrence du défaut ne peut être prise.

Dans cette section, nous commençons par définir la notion de diagnosticabilité d'un langage temporisé, étant donnée une partition sur l'ensemble des défauts. Ensuite, nous présentons une méthode systématique permettant de vérifier la diagnosticabilité d'un langage accepté par un automate temporisé soumis aux hypothèses considérées dans notre travail. Enfin, nous présentons un théorème permettant de lier la notion de diagnosticabilité avec l'application de la méthode de vérification proposée.

Algorithm 3 Procédure de synthèse du diagnostiqueur

Entrées : : Automate temporisé du système model $A = (Q, X, \Sigma, E, q_0, Q^f, I)$.

Sorties : Automate temporisé du diagnostiqueur $D = (Q^d, X, \Sigma_o, E^d, q_0^d, Q_f^d, I^d)$.

$q_0^d \leftarrow (\{(q_0, N)\}, z_0)$
$SOMMETS\_A\_TRAITER \leftarrow \{q_0^d\}$
$Q^d \leftarrow \{q_0^d\}$, $E^d \leftarrow \{\}$
tantque $SOMMETS\_A\_TRAITER \neq \{\}$ **faire**
  retirer un sommet q^d de $SOMMETS\_A\_TRAITER$
  calculer les états d'ombre de q^d, $\acute{E}TATS\_OMBRE = \mathcal{UR}(q^d)$.
  déterminer $q_f^d = (\{(q_i, \phi_i), i \in \{1, \ldots, k\}\}, z_f) | (q_i, \phi_i, z_i) \in \acute{E}TATS\_OMBRE$ et $q_i \in Q_f$, $\forall i \in \{1, \cdots\}$, et $z_f = z_1 \wedge \cdots \wedge z_k)\}$
  si q_f^d n'est pas vide **alors**
    déterminer $\tau = min\{d \in \{1, \ldots, K+1\} | \forall (q, z, \phi) \in \acute{E}TATS\_OMBRE, (\langle z \wedge (y < d)\rangle = \emptyset) \Rightarrow q \in Q_f\}$
    ajouter q_f^d à Q^d et Q_f^d,
    ajouter une transition urgente $q^d \xrightarrow{\tau} q_f^d$
  finsi
  pour tout $\sigma \in \Sigma_o$ **faire**
    déterminer l'ensemble $\acute{E}TATS\_FRANCHISSABLES^{\sigma} = \{(q_i, \eta_i, \phi_i) | (q_i, z_i, \phi_i) \in \acute{E}TATS\_OMBRE, q_i \xrightarrow{\sigma, g, Y} \tilde{q}_i \in E,\ and\ \eta_i = (z_i \wedge g) \neq \emptyset\}$
    tantque $\exists (q_i, \eta_i, \phi_i), (q_j, \eta_j, \phi_j) \in \acute{E}TATS\_FRANCHISSABLES^{\sigma} \mid (\eta_i \wedge \eta_j \neq \emptyset)$ and $(\eta_i \neq \eta_j)$ **faire**
      Retirer (q_i, η_i, ϕ_i) et (q_j, η_j, ϕ_j) de $\acute{E}TATS\_FRANCHISSABLES^{\sigma}$.
      Ajouter $\{(q_i, \langle \eta_i \wedge \eta_j \rangle, \phi_i), (q_j, \langle \eta_i \wedge \eta_j \rangle, \phi_j), (q_j, \langle \neg \eta_i \wedge \eta_j \rangle, \phi_j), (q_i, \langle \eta_i \wedge \neg \eta_j \rangle, \phi_i)\}$ à $\acute{E}TATS\_FRANCHISSABLES^{\sigma}$.
    fin tantque
    $ZONES\_TRAIT\acute{E}ES \leftarrow \{\}$
    pour tout zone $\eta \in \mathcal{C}(X) | \exists (q, \eta, \phi) \in \acute{E}TATS\_FRANCHISSABLES^{\sigma}$ et $\eta \notin ZONES\_TRAIT\acute{E}ES$ **faire**
      ajouter η à $ZONES\_TRAIT\acute{E}ES$
      créer un nouveau sommet $\tilde{q}^d = (\{(\tilde{q}_i, \phi_i), i \in \{1, \ldots, k\}\}, \tilde{z})$, où $\tilde{z} = \eta[Y \leftarrow 0]$, $(q_i, \eta, \phi_i) \in \acute{E}TATS\_FRANCHISSABLES^{\sigma}$, et $q_i \xrightarrow{\sigma, g, Y} \tilde{q}_i \in E$.
      ajouter le sommet $\tilde{q}^d$ à Q^d et $SOMMETS\_A\_TRAITER$, si $\tilde{q}^d \notin Q^d$.
      ajouter une transition $q^d \xrightarrow{\sigma, \eta, Y} \tilde{q}^d$ à E^d.
    fin pour
  fin pour
fin tantque

4.5.1 La diagnosticabilité des langages temporisés

Soit L un langage temporisé à temps-divergent sur un alphabet Σ. Nous supposons que $\Sigma = \Sigma_o \cup \Sigma_{uo}$, où Σ_o, respectivement Σ_{uo}, désigne l'ensemble des événements observables, respectivement non observables. Par ailleurs, nous supposons qu'une partition de m modes de défauts $\Sigma_f = F_1 \cup \cdots \cup F_m$ est définie sur l'ensemble des défauts $\Sigma_f \subseteq \Sigma_{uo}$.

Dans le cadre de notre étude de la diagnosticabilité des langages temporisés, nous adoptons une notation alternative pour représenter les traces temporisées. Cette notation nous permet de faciliter la représentation des traces où le temps s'écoule sans l'occurrence d'événements. En effet, une *trace temporisée* est définie comme une séquence, finie ou infinie, d'événements alternés avec des nombre réels positifs : $\delta_0, \sigma_1, \delta_1, \sigma_2, \delta_2, \ldots, \sigma_k, \delta_k \ldots$, où les σ_i, pour $i \geq 1$, sont des éléments de Σ et $\delta_i \in \mathbb{R}_+$, correspond au temps écoulé entre les occurrences des événements successifs σ_i et σ_{i+1}. Une trace est dite *valide* si : (1) elle ne contient pas deux événements successifs et (2) si elle contient deux délais successifs, alors tout les éléments qui suivent ces délais sont aussi des délais.

Exemple 4.5.1
Nous présentons dans la suite quelques exemples de traces valides et non valides.

- la trace "$\omega = 5, a, 6, b, 3.1, c$" est une trace valide. Cette trace est équivalente à "$\omega = (a, 5)(b, 6)(c, 3.1)$" selon la notation utilisée dans les parties précédentes ;
- la trace "$\omega = 5, a, 6, b, 3.1, 5, c$" n'est pas une trace valide. En effet, on ne doit pas avoir deux délais successifs, suivis par un événement. Une notation valide de cette trace pourra être "$\omega = 5, a, 6, b, 8.1, c$" ;
- la trace "$\omega = 5, a, 6, b, 6.5, c, 1, 1, 1, 1, 1 \ldots$" est une trace valide. Elle permet de représenter l'écoulement infini du temps après l'occurrence de l'événement c. Il faut noter que cette trace ne peut être représentée en utilisant la notation basée sur une séquence de couples (événement,délai).

Nous notons par P_{Σ_o} l'opérateur permettant de projeter une trace temporisée sur l'ensemble des événements observables. Pour une meilleure lisibilité, nous pouvons simplement noter cet opérateur par P. Nous donnons dans la suite quelques exemples de projections observables de traces temporisées. Nous considérons que les événements a, b et c sont les uniques événements observables.

Exemple 4.5.2

- pour "$\omega = 2, a, 2, u, 6.6, b, 2.1, f, 2, c$", "$P(\omega) = 2, a, 8.6, b, 4.1, c$" ;
- pour "$\omega = 2, a, 3, f, 6, b, 1, 1, 1, \ldots$", "$P(\omega) = 2, a, 9, b, 1, 1, 1, \ldots$".

Nous associons à chaque mode de défauts $F_i, i \in \{1, \ldots, m\}$, un sous-langage L_i de L, tel que chaque trace temporisée dans L_i contient au moins un défaut de l'ensemble F_i. Étant donné un nombre réel positif Δ, nous désignons par L_i^{Δ}, $i \in \{1, \ldots, m\}$, le sous-langage de L_i, où plus que Δ t.u. se sont écoulées après l'occurrence d'un défaut de l'ensemble F_i, dans chaque trace de L_i^{Δ}. En effet, étant donnée une trace temporisée $\omega \in L_i^{\Delta}$, $\omega = \delta_0, \sigma_1, \delta_1, \sigma_2, \ldots, \sigma_n, \delta_n \ldots$, nous avons $\sum_{k \geq j} \delta_k \geq \Delta$, où σ_j correspond à un événement de l'ensemble F_i.

Exemple 4.5.3
Nous considérons dans cet exemple un ensemble d'événements $\Sigma = \{a, b, c, f_1, f_2\}$ et une partition de défauts $\Sigma_f = F_1 \cup F_2$, où $F_1 = \{f_1\}$ et $F_2 = \{f_2\}$. Par ailleurs, nous considérons les traces temporisées "$\omega = 4, a, 6, b, 1, f_1, 3, c$" ; "$\omega' = 1, a, 2, f_2, 6, b, 4, c$" et "$\omega'' = 1, a, 5, f_2, 7, b, 9, c, 1, 1, \ldots$" :

- il est clair que $\omega \in L_1$ et $\omega' \in L_2$;
- pour $\Delta = 10$, nous avons $\omega' \in L_2^{\Delta}$ et $\omega \notin L_1^{\Delta}$;
- pour $\Delta = 2$, nous avons $\omega' \in L_2^{\Delta}$ et $\omega \in L_1^{\Delta}$;
- pour tout $\Delta \geq 0$, nous avons $\omega'' \in L_2^{\Delta}$.

Dans ce qui suit, nous présentons une définition formelle pour la diagnosticabilité d'un langage temporisé.

Définition 28. Soit L un langage temporisé à temps-divergent. Le langage L est dit *diagnosticable* par rapport à une partition de m modes de défauts, $\Sigma_f = F_1 \cup \cdots \cup F_m$, si :

$$(\forall i \in \{1, \ldots, m\})(\exists \Delta \in \mathbb{R}_+)(\forall \omega \in L_i^{\Delta})$$
$$\forall \tilde{\omega} \in L,\ (P(\omega) = P(\tilde{\omega})) \Longrightarrow \tilde{\omega} \in L_i.$$

Cette définition stipule qu'un langage temporisé à temps-divergent est diagnosticable si pour toute paire de traces, la première contenant un défaut de l'ensemble F_i, $i \in \{1, \ldots, m\}$ et la seconde ne contenant aucun défaut de F_i, il existe un délai Δ, tel que les deux traces auront des projections observables différentes au bout de l'écoulement de Δ t.u. à partir de l'occurrence du défaut de F_i.En effet, la condition de diagnosticabilité d'un langage temporisé garantit l'identification de tout défaut de l'ensemble F_i, $i \in \{1, \ldots, m\}$ au bout d'un temps fini de son occurrence, étant donnée la séquence d'événements observable qui suit ce défaut.

Dans la suite, nous présentons un exemple dans lequel nous étudions la diagnosticabilité du langage accepté par un simple automate temporisé.

Exemple 4.5.4
Dans cet exemple, nous étudions la diagnosticabilité du language temporisé

à **temps-divergent** accepté l'automate temporisé décrit dans la figure 4.20. Nous supposons que les événements a et b sont observables et que les événements f_1 et f_2 correspondent à des événements de défauts non observables. Par ailleurs, nous adoptons la partition de défauts suivante : $\Sigma_f = F_1 \cup F_2$, où $F_1 = \{f_1\}$ et $F_2 = \{f_2\}$. Puisque les sommets 3, 6 et 9 sont finaux, leur contrainte d'invariant est égale à $vrai$. Le reste des sommets est associé à l'invariant $x \leq 10 \wedge y \leq 10$.

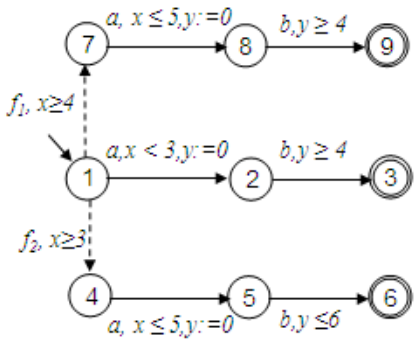

Fig. 4.20 – Etude de la diagnosticabilité d'un automate temporisé

Nous pouvons remarquer l'existence de deux traces temporisées à temps divergent (donc admissibles) "$\omega_1 = 4.1, f_1, 0.8, a, 5, b, 1, 1, \ldots$" et "$\omega_2 = 3.4, f_2, 1.5, a, 5, b, 1, 1, \ldots$", acceptées par ce modèle telles que : "$P(\omega_1) = P(\omega_2) = 4.9, a, 5, b, 1, 1, \ldots$", $\omega_1 \in L_1$ et pour tout $\Delta \geq 0$, $\omega_2 \in L_2^{\Delta}$. Puisque la définition de diagnosticabilité définie ci-dessus est violée, nous pouvons déduire que le langage à temps-divergent, accepté par l'automate temporisé considéré, n'est pas diagnosticable.

Dans un deuxième cas, nous considérons une autre partition de défauts : $\Sigma_f = F_1 = \{f_1, f_2\}$. Le modèle vérifie alors la condition de diagnosticabilité. En effet, nous pouvons discriminer les traces contenant un défaut de l'ensemble F_1, à partir des événements observables qui suivent ce défaut, en se basant plus précisément sur la date d'occurrence de l'événement a. Ainsi, si l'événement a est observé avant l'écoulement de 3 u.t., à partir de l'état initial, alors nous pouvons déduire qu'aucun défaut ne s'est produit. Dans le cas contraire (a est observé après l'écoulement de 3 u.t.), nous déduisons qu'un défaut de l'ensemble F_1 s'est produit, sans déterminer exactement lequel des deux événements f_1 ou f_2. En conclusion, le language à temps-divergent accepté par ce modèle est diagnosticable en considérant cette partition de défauts et $\Delta = 3$.

Remarque 5. Nous soulignons que la notion de diagnosticabilité s'applique uniquement aux traces temporisées admissibles ; i.e., des traces qui peuvent être étendues à

des traces à temps-divergent. La divergence du temps dans une exécution du système est une propriété naturellement satisfaite lorsqu'il s'agit d'un système réel. Ainsi, les exécutions dans lesquelles le temps converge (exécutions zénon) ou s'arrête de progresser sont généralement dues à l'existence d'erreurs de modélisation. Dans le modèle considéré dans notre approche, les hypothèses retenues sur le modèle, plus précisément l'hypothèse $\mathcal{H}_4$, peuvent conduire à de telles situations, où le temps ne progresse plus au delà d'une valeur déterminée. Toutefois, ces traces ne seront pas considérées dans notre démarche de vérification de diagnosticabilité, qui considère uniquement les traces admissibles.

4.5.2 Vérification de la diagnosticabilité

Nous proposons dans la suite une approche systématique pour la vérification de la diagnosticabilité d'un langage temporisé à temps-divergent. Ce langage est accepté par un automate temporisé A soumis aux spécifications et aux hypothèses énumérées dans la section 4.2. Nous commençons par définir un modèle indispensable permettant la caractérisation les conditions de vérification de la diagnosticabilité, appelé *automate symbolique compagnon* du diagnostiqueur. Il s'agit d'un automate temporisé, généré par la procédure de synthèse du diagnostiqueur, au même moment que le diagnostiqueur. Ensuite, nous illustrons notre méthode permettant de vérifier la notion de diagnosticabilité. Cette méthode repose sur la vérification de quelques propriétés structurelles sur l'automate temporisé du diagnostiqueur ainsi que sur son automate symbolique compagnon. Enfin, un théorème liant la diagnosticabilité d'un modèle et la vérification de ces conditions est formalisé.

4.5.2.1 Automate symbolique compagnon

L'automate symbolique compagnon est un automate temporisé construit simultanément avec l'automate temporisé du diagnostiqueur. Chaque sommet de cet automate correspond à un état symbolique représentant un ensemble d'états d'entrée ou un ensemble d'états d'ombre. Intuitivement, l'automate symbolique compagnon correspond au dépliage de l'automate temporisé du système. En effet, l'algorithme de construction du diagnostiqueur établit implicitement une analyse d'accessibilité en avant pour déterminiser les transitions franchissables sur les événements observables. Les états d'entrée et les états d'ombre résultants de cette construction constituent les éléments de l'automate compagnon. Nous référons l'appellation automate symbolique compagnon au fait que les sommets de cet automate correspondent à des états symboliques construits lors de la synthèse du diagnostiqueur.

Nous allons décrire un exemple permettant de présenter le modèle automate symbolique compagnon du diagnostiqueur introduit dans l'exemple 4.4.4.

Exemple 4.5.5
L'automate temporisé de la figure 4.21(b) représente l'automate symbolique compagnon du diagnostiqueur décrit dans la figure 4.21(a).

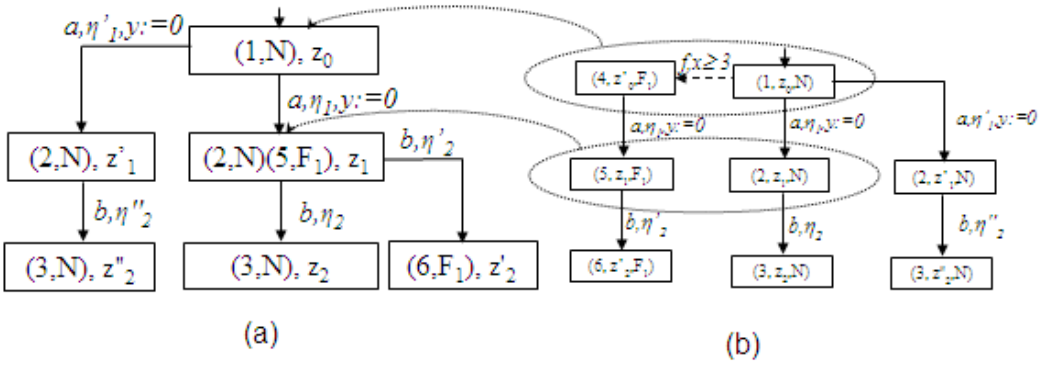

Fig. 4.21 – (a) Un diagnostiqueur et (b) son automate symbolique compagnon

Il est clair, d'après la figure 4.21(b), que l'automate compagnon du diagnostiqueur correspond à un dépliage du modèle automate temporisé du système illustré dans la figure 4.17. Ce modèle est assez semblable à un automate de zones construit en effectuant une analyse en avant d'un automate temporisé (Bengtsson et Yi., 2004). A chaque chemin dans l'automate temporisé du système est associé un ensemble de chemins dans l'automate compagnon, d'où découle l'équivalence entre ces deux modèles. Nous notons aussi que l'automate du diagnostiqueur peut être retrouvé à partir de son automate symbolique compagnon, en regroupant certain sommet de ce modèle, comme il est illustré dans notre exemple. La procédure complète permettant la synthèse du diagnostiqueur et de son automate symbolique compagnon est décrite dans l'algorithme enrichi présenté dans l'annexe A. Dans cet algorithme, les modifications apportées à la procédure initiale présentée dans l'algorithme 3, sont marquées en caractères gras. Dans la suite de notre travail, l'automate temporisé compagnon sera désigné par le septuplet $S = (Q^s, X, \Sigma, E^s, q_0^s, Q_f^s, I^s)$, où $Q^s \subseteq Q \times \mathcal{C}(X) \times \Phi$ et $q_0^s = (q_0, z_0, \mathcal{N})$.

Dans notre exemple, les deux chemins $(1, z_0, \mathcal{N}) \xrightarrow{a,\eta_1,y:=0} (2, z_1, \mathcal{N}) \xrightarrow{b,\eta_2} (3, z_2, \mathcal{N})$ et $(1, z_0, \mathcal{N}) \xrightarrow{a,\eta_1',y:=0} (2, z_1', \mathcal{N}) \xrightarrow{b,\eta_2''} (3, z_2'', \mathcal{N})$, dans l'automate symbolique compagnon correspondent au chemin $1 \xrightarrow{a,x\leq 10,y:=0} 2 \xrightarrow{b,x\geq 12} 3$ dans l'automate temporisé du système. En effet, l'ensemble des traces acceptées par le chemin dans l'automate temporisé du système est égale à l'ensemble de traces temporisées acceptées par ses deux correspondants chemins dans l'automate symbolique compagnon. En généralisant cette propriété structurelle, nous pouvons déduire le lemme suivant :

Lemme 1. Soient A un automate temporisé vérifiant les hypothèses considérées dans notre travail, D l'automate temporisé de son diagnostiqueur et A l'automate symbolique compagnon. Soient $L(A), L(S)$ et $L(D)$ les langages temporisés acceptés respectivement par les automates temporisés A, S et D, alors :

1. $L(S) = L(A)$;
2. $L(D) = P(L(A))$.

Ce lemme établit une équivalence entre le langage accepté par l'automate temporisé A du système et les langages acceptés par le diagnostiqueur D et son automate symbolique compagnon S. Ce lemme découle de la construction des modèles du diagnostiqueur et son automate compagnon. Il permet d'un coté de traduire l'équivalence entre le langage accepté par le modèle du système et celui accepté par l'automate compagnon, et d'un autre coté entre le langage accepté par le diagnostiqueur et la projection observable du langage accepté par son compagnon d'un autre.

Dans la suite, nous proposons un lemme qui découle de la construction du diagnostiqueur. Ce lemme établit que, pour chaque paire d'états d'entrée appartenant à un sommet du diagnostiqueur et ayant la même valuation d'horloges, il existe une paire de traces temporisées ayant la même projection observable et faisant évoluer l'automate temporisé du système vers ces deux états.

Lemme 2. Soit q^d un sommet du diagnostiqueur alors :

1. pour chaque élément $(q, \phi) \in Dis(q^d)$, $v \in Z(q^d)$, il existe une exécution de A sur une trace temporisée ω, telle que $(q_0, v_0) \overset{\omega}{\rightsquigarrow} (q, v)$;
2. s'il existe une paire d'éléments $(q, \phi), (\tilde{q}, \tilde{\phi}) \in Dis(q^d)$, alors pour toute valuation d'horloges $v \in Z(q^d)$, il existe une paire d'exécutions de A sur les traces temporisées ω et $\tilde{\omega}$, respectivement, telle que $(q_0, v_0) \overset{\omega}{\rightsquigarrow} (q, v)$, $(q_0, v_0) \overset{\tilde{\omega}}{\rightsquigarrow} (\tilde{q}, v)$ et $P(\omega) = P(\tilde{\omega})$.

Preuve : voir annexe B.1.

Dans la suite, nous présentons un lemme qui découle du lemme 2. Il stipule que pour chaque sommet F_i-incertain du diagnostiqueur, il existe une paire de traces acceptées par l'automate temporisé du système et ayant la même projection observable, la première contient un défaut de l'ensemble F_i et la seconde ne contient aucun défaut de l'ensemble F_i. Par ailleurs, l'exécution sur la projection observable de ces deux traces fait évoluer l'automate temporisé du diagnostiqueur vers le sommet F_i-incertain en question.

Lemme 3. Soit q^d un sommet F_i-incertain du diagnostiqueur, $i \in \{1, \ldots, m\}$ et soient $(q, \phi), (\tilde{q}, \tilde{\phi}) \in Dis(q^d)$, où $\phi = \mathcal{F}_i$ et $\tilde{\phi} \neq \mathcal{F}_i$, alors, pour toute valuation $v \in Z(q^d)$, il existe une paire de traces ω et $\tilde{\omega}$, telle que :

- $(q_0, v_0) \overset{\omega}{\rightsquigarrow} (q, v)$ et $(q_0, v_0) \overset{\tilde{\omega}}{\rightsquigarrow} (\tilde{q}, v)$;
- $P(\omega) = P(\tilde{\omega})$;

- $\omega \in L_i$ et $\tilde{\omega} \in L - L_i$.

Preuve : immédiate à partir du lemme 2 et le principe de propagation des étiquettes de diagnostic au cours de la construction du diagnostiqueur.

Le lemme suivant découle aussi du lemme 2. Il établit que lorsque le diagnostiqueur évolue vers un sommet F_i-certain sur une trace d'événements observables générés par le système, alors toute trajectoire possible du système, ayant le même comportement observable que cette trace, comporte au moins un défaut de l'ensemble F_i.

Lemme 4. Soit q^d un sommet F_i-certain du diagnostiqueur, $i \in \{1, \ldots, m\}$, atteignable à travers une exécution sur une trace ω. Alors, pour toute trace $\tilde{\omega} \in L$, telle que $P(\tilde{\omega}) = \omega$, nous avons $\tilde{\omega} \in L_i$.

Preuve : immédiate à partir du lemme 2 et le principe de propagation des étiquettes de diagnostic au cours de la construction du diagnostiqueur.

4.5.2.2 Conditions pour la vérification de diagnosticabilité

Nous présentons dans cette partie les conditions permettant de garantir la diagnosticabilité du (langage accepté par le) modèle considéré dans notre travail. Ces conditions sont vérifiées sur la structure du diagnostiqueur et son automate symbolique compagnon. D'abord, nous définissons quelques notions nécessaires pour caractériser les conditions de diagnosticabilité. Ensuite, nous détaillons ces conditions. Enfin, un théorème liant formellement la vérification de ces conditions et la définition du diagnosticabilité est formalisé.

Définition 29. Un ensemble de sommets F_i-incertain du diagnostiqueur $\{q_1^d, q_2^d, \ldots, q_n^d\}$, pour $i \in \{1, \ldots, m\}$, constitue un cycle *F_i-indéterminé* du diagnostiqueur D, si les conditions suivantes sont vérifiées :

1. $q_1^d \xrightarrow{\sigma_1, g_1, Y_1} q_2^d \ldots q_{n-1}^d \xrightarrow{\sigma_{n-1}, g_{n-1}, Y_{n-1}} q_n^d \xrightarrow{\sigma_n, g_n, Y_n} q_1^d$ correspond à un cycle de sommets dans D, avec $\sigma_1, \ldots, \sigma_n \in \Sigma_o$.

2. Il existe un cycle dans l'automate symbolique compagnon S, dont les sommets contiennent l'étiquette $\mathcal{F}_i$ et une partie ou tous les sommets[1] de ce cycle sont inclus dans les sommets du cycle du diagnostiqueur. Ce cycle est appelé cycle *F_i-défaillant*, noté $\mathcal{C}_i$.
3. Il existe un cycle dans l'automate symbolique compagnon S, dont les sommets contiennent des étiquettes différentes de $\mathcal{F}_i$ et une partie ou tous les sommets de

[1] les sommets accessibles sur des transitions sur des événements observables.

ce cycle sont inclus dans les sommets du cycle du diagnostiqueur. Ce cycle est appelé cycle F_i-*sûr*, noté $\overline{\mathcal{C}_i}$.

Nous présentons dans la suite les conditions de diagnosticabilité du langage temporisé accepté par l'automate temporisé A, vérifiant les conditions de notre travail.

Conditions de diagnosticabilité :

C_1 le diagnostiqueur D ne contient aucun cycle F_i-indéterminé ;

C_2 aucun sommet final dans D n'est F_i-incertain.

Intuitivement, ces conditions permettent de garantir l'absence de toute paire d'exécutions sur deux traces ayant la même projection observable, la permière contient un défaut de l'ensemble F_i et la seconde ne contenant aucun défaut de F_i. De telles exécutions peuvent exister si la condition C_1 n'est pas vérifiée. En effet, la première exécution évolue dans le cycle F_i-défaillant et la deuxième exécution évolue dans le cycle F_i-sûr, tout en ayant la même projection observable pour n'importe quelle durée d'exécution. De même si la deuxième condition n'est pas vérifiée ; i.e., le diagnostiqueur comporte un sommet final et F_i-certain, alors il existe deux traces finies ayant la même projection observable, l'une contient un défaut de F_i et l'autre non. De plus, le temps peut s'écouler indéfiniment dans la suite de ces exécutions sans l'occurrence d'autres événements observables. En effet, il serait impossible de distinguer entre ces deux exécutions à partir de l'ensemble des événements observées, pour toute durée d'écoulement du temps.

Dans la suite, nous considérons un exemple illustrant le cas d'un diagnostiqueur comportant un cycle F_i-indéterminé.

Exemple 4.5.6

Le diagnostiqueur du système de chauffage de liquides présenté dans la figure 4.13 ne comporte pas de cycles de sommets F_i-incertain. En effet, le modèle du système, à partir duquel le diagnostiqueur a été construit, est diagnosticable. Nous proposons dans la suite d'introduire des modifications sur le modèle de la figure 4.12. Le modèle obtenu est présenté dans la figure 4.22. Dans la nouvelle version du modèle, nous pouvons remarquer que la transition (**évac & fuite**) $\xrightarrow{vide, 15 \leq x \leq 22, x:=0}$ (**rempl & fuite**) admet une condition de garde différente que celle définie dans le modèle original.

Nous constatons que l'introduction de cette simple modification rend ce modèle non diagnosticable. En effet, nous pouvons remarquer à partir du modèle du diagnostiqueur et de son automate symbolique compagnon, représentés respectivement dans les figures 4.23 et 4.24, l'existence d'un cycle F_1-incertain. Si nous considérons une exécution du diagnostiqueur sur la trace d'événements observables $((rempli, 45)(évacuer, 60)(vide, 21))^*$, deux trajectoires peuvent exister dans ce cas expliquant la trace ob-

servé. En effet, le système peut évoluer dans le cycle $\mathcal{C}_1$ sur la trace $(fuite, 0)((rempli, 45)(évacuer, 60)(vide, 21))^*$ ou dans le cycle $\overline{\mathcal{C}_1}$ sur la trace $((rempli, 45)(évacuer, 60)(vide, 21))^*$. Ainsi, il n'est pas possible de différencier entre les projections observables de ces deux comportements, pour toute la durée d'observation du système.

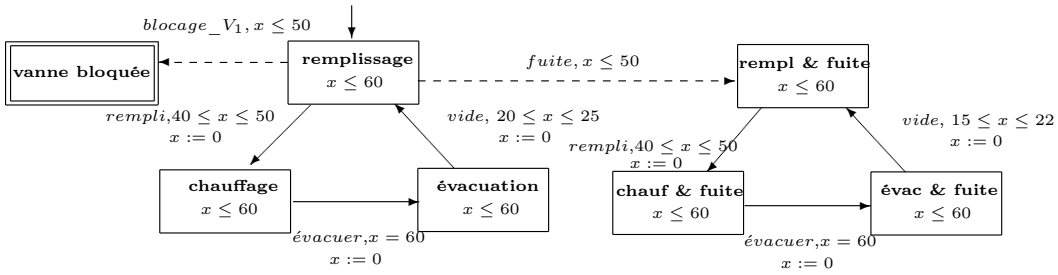

Fig. 4.22 – *Un modèle non diagnosticable.*

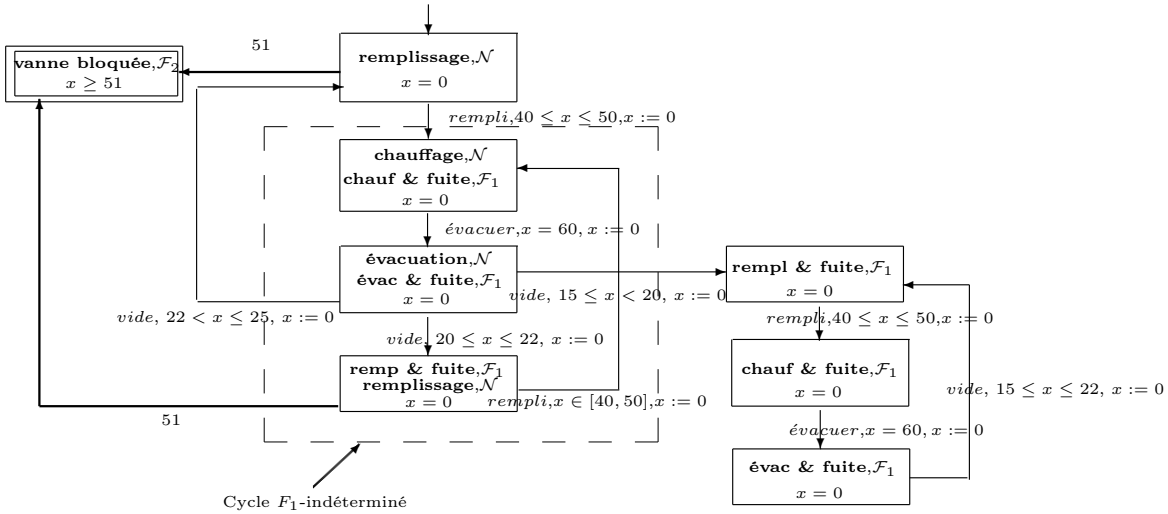

Fig. 4.23 – *Un diagnostiqueur comportant un cycle F_1-indéterminé*

Le théorème suivant établit une condition nécessaire et suffisante pour la diagnosticabilité d'un langage temporisé accepté par un automate temporisé, vérifiant les hypothèses de notre approche.

Théorème 1 (Diagnosticabilité)**.** Soit A un automate temporisé vérifiant les spécifications et les hypothèses énumérées dans la section 4.2 et soit D le diagnostiqueur obtenu par l'application de notre procédure de synthèse. Le langage temporisé L accepté par A est diagnosticable, si et seulement si, pour tout $i \in \{1, \ldots, m\}$:

1. D ne contient aucun cycle F_i-indéterminé ;
2. D ne contient aucun sommet à la fois final et F_i-incertain.

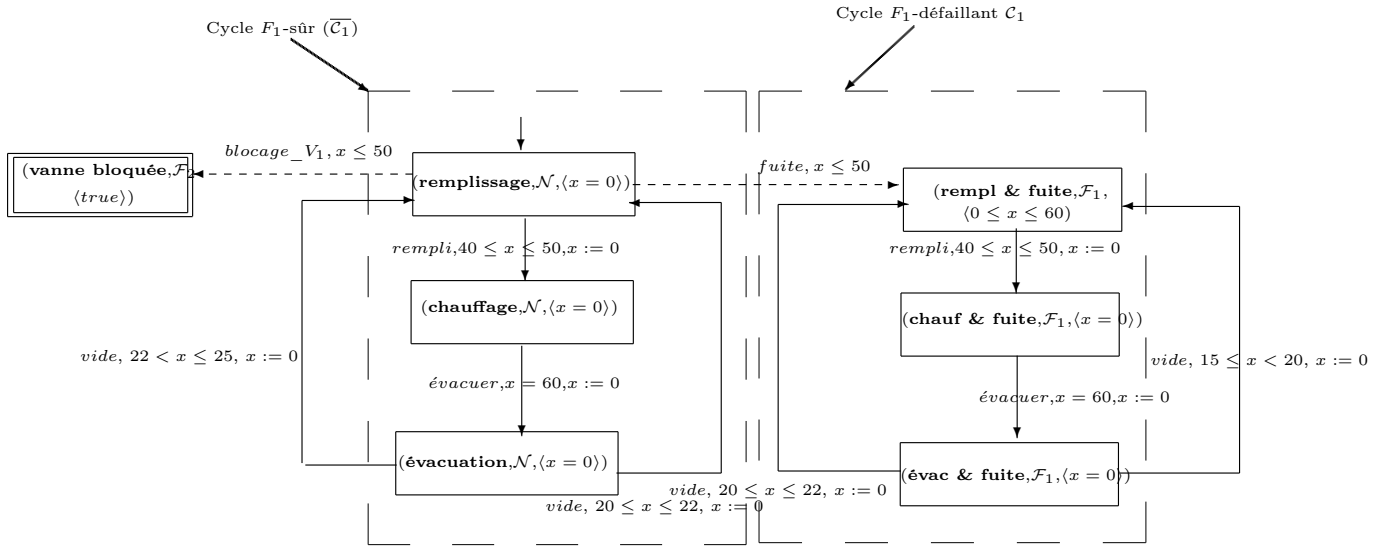

Fig. 4.24 – *Un automate symbolique compagnon comportement deux cycles $\mathcal{C}_1$ et $\overline{\mathcal{C}_1}$*

Afin de prouver ce théorème, nous commençons par présenter deux lemmes établissant quelques propriétés sur les cycles F_i-indéterminé. Nous prouvons que, étant donné un cycle F_i-indéterminé dans le diagnostiqueur, il existe pour tout entier $\alpha \geq 0$, une paire de traces appartenant au langage L et ayant la même projection observable, où l'exécution de S sur la première trace le fait évoluer α fois dans le cycle $\mathcal{C}_i$, et l'exécution de S sur la deuxième trace le fait évoluer dans le cycle $\overline{\mathcal{C}_i}$. Afin de prouver l'existence de ces deux traces, nous introduisons un premier lemme qui permet la construction d'une partie de ces traces en raisonnant sur chaque sommet et son prédécesseur dans le cycle F_i-indéterminé . Ensuite, le résultat de ce premier lemme sera utilisé dans le second pour la construction de la paire de traces parcourant les cycles $\mathcal{C}_i$ et $\overline{\mathcal{C}_i}$.

Lemme 5. Soit q^d_{n+1} un sommet appartenant à un cycle F_i-indéterminé et soient $(q_{n+1}, \mathcal{F}_i), (\tilde{q}_{n+1}, \tilde{\phi}_{n+1}) \in Dis(q^d_{n+1})$, tels que $\tilde{\phi}_{n+1} \neq \mathcal{F}_i$, $(q_{n+1}, \mathcal{F}_i, z_{n+1}) \in \mathcal{C}_i$ et $(\tilde{q}_{n+1}, \tilde{\phi}_{n+1}, z_{n+1}) \in \overline{\mathcal{C}_i}$, où $z_{n+1} = Z(q^d_{n+1})$. Nous notons par q^d_n le prédécesseur de q^d_{n+1} dans le cycle F_i-indéterminé, sur la transition $q^d_n \xrightarrow{\sigma_{n+1}, g_{n+1}, Y_{n+1}} q^d_{n+1} \in E^d$. Alors, pour toute valuation $v_{n+1} \in z_{n+1}$, il existe une valuation $v_n \in z_n = Z(q^d_n)$, une paire d'éléments $(q_n, \mathcal{F}_i), (\tilde{q}_n, \tilde{\phi}_n) \in Dis(q^d_n)$ et une paire de traces temporisées ω_{n+1} et $\tilde{\omega}_{n+1}$, telles que :

- $(q_n, \mathcal{F}_i, z_n) \in \mathcal{C}_i$ et $(\tilde{q}_n, \tilde{\phi}_n, z_n) \in \overline{\mathcal{C}_i}$;
- $(q_n, v_n) \overset{\omega_{n+1}}{\rightsquigarrow} (q_{n+1}, v_{n+1})$ et $(q_n, v_n) \overset{\tilde{\omega}_{n+1}}{\rightsquigarrow} (q_{n+1}, v_{n+1})$;
- $P(\omega_{n+1}) = P(\tilde{\omega}_{n+1}) =$ "$d.\sigma_{n+1}$", où $d \in \mathbb{R}_+$.

Preuve : voir annexe B.1.

Lemme 6. Soit q^d_n un sommet du diagnostiqueur appartenant à un cycle F_i-indéterminé et $(q_n, \mathcal{F}_i), (\tilde{q}_n, \tilde{\phi}_n) \in Dis(q^d)$, tels que $\tilde{\phi}_n \neq \mathcal{F}_i$, $(q^p_n, \mathcal{F}_i, z_n) \in \mathcal{C}_i$, $(\tilde{q}^q_n, \tilde{\phi}^q_n, z_n) \in \overline{\mathcal{C}_i}$ et $z_n = Z(q^d_n)$.

Alors, étant donné un entier naturel $\alpha \geq 0$ et $\forall v^0_{n,p} \in z_n$, il existe une valuation d'horloges $v^\alpha_{n,p} \in z_n$ et une paire de traces temporisées $\omega^\alpha, \tilde{\omega}^\alpha$ telles que :

1. L'exécution $(q^p_n, v^\alpha_{n,p}) \overset{\omega^\alpha}{\rightsquigarrow} (q^p_n, v^0_{n,p})$ fait évoluer S α fois dans le cycle $\mathcal{C}_i$,
2. L'exécution $(\tilde{q}^{q'}_n, v^\alpha_{n,p}) \overset{\tilde{\omega}^\alpha}{\rightsquigarrow} (\tilde{q}^q_n, v^0_{n,p})$ fait évoluer S dans le cycle $\overline{\mathcal{C}_i}$,
3. $P(\omega^\alpha) = P(\tilde{\omega}^\alpha)$.

Preuve : voir annexe B.1.

***Preuve du théorème :* 1**

Nécessité :
Nous prouvons la nécessité par contradiction. Deux cas seront étudiés :

Premier cas : le diagnostiqueur contient un sommet F_i-incertain et final $q^d \in Q^f_d$. Alors, $\exists (q, \mathcal{F}_i), (\tilde{q}, \tilde{\phi}) \in Dis(q^d)$, $i \in \{1, \ldots, m\}$ and $\tilde{\phi} \neq \mathcal{F}_i$, and $q, \tilde{q} \in Q^f$.

D'après le lemme 3, $\forall v \in Z(q^d)$, il existe une paire de traces temporisées : ω et $\tilde{\omega}$, telle que :

- $P(\omega) = P(\tilde{\omega})$,
- $(q_0, v_0) \overset{\omega}{\rightsquigarrow} (q, v)$ et $(q_0, v_0) \overset{\tilde{\omega}}{\rightsquigarrow} (\tilde{q}, v)$.
- $\omega \in L_i$ et $\tilde{\omega} \in L - L_i$.

Nous désignons par t_f la date de la première occurrence d'un défaut de l'ensemble F_i dans ω. Puisque les sommets q et $\tilde{q}$ sont finaux ; i.e., les invariants sur les sommets sont non bornés (égales à $vrai$), alors chaque exécution atteignant ces sommets vont y séjourner indéfiniment sans pouvoir évoluer vers un d'autres sommets. Alors, étant données deux valeurs $\Delta \in \mathbb{R}_+$ et $d > t_f + \Delta$, il existe deux traces $\omega_e, \tilde{\omega}_e \in L$ telles que $\omega_e =$"$\omega.d$", $\tilde{\omega}_e =$ "$\tilde{\omega}.d$". Puisque $P(\omega_e) = P(\tilde{\omega}_e)$, $\omega_e \in L_i$ et $\tilde{\omega}_e \in L - L_i$, nous concluons que notre hypothèse de départ est contredite et par conséquent L n'est pas diagnosticable.

Second cas : il existe un cycle F_i-indéterminé dans D. En effet, il existe un cycle de sommets F_i-incertain dans D, auquel correspond un cycle F_i-défaillant ; i.e., $\mathcal{C}_i$, et un cycle F_i-sûr ; i.e., $\overline{\mathcal{C}_i}$, dans l'automate S.

Puisque l'automate temporisé A est fortement non-zeno, alors chaque exécution cyclique admet une durée d'exécution supérieure à une valeur $\delta > 0$. Puisque chaque cycle dans l'automate temporisé S correspond à au moins un cycle dans A, alors la durée écoulée par chaque exécution évoluant dans le cycle $\mathcal{C}_i$ est supérieure à δ u.t.. Nous désignons par Δ une valeur arbitrairement large et $\alpha = \lfloor \Delta/\delta \rfloor + 1$.

Soit q^d_k un élément du cycle F_i-indéterminé. Puisque q^d_k is F_i-incertain, alors il existe $(q_k, \mathcal{F}_i)$ et $(\tilde{q}_k, \tilde{\phi}_k \neq \mathcal{F}_i) \in Dis(q^d)$, tels que $(q_k, \mathcal{F}_i, Z(q^d_k)) \in \mathcal{C}_i$ et $(\tilde{q}_k, \tilde{\phi}_k, \neq \mathcal{F}_i, Z(q^d_k)) \in$

$\overline{\mathcal{C}_i}$. Soit v_k^0 une valuation dans $Z(q_k^d)$. D'après le lemme 6, il existe une valuation d'horloges $v_k^\alpha \in Z(q_k^d)$ et une paire de traces temporisées $\omega^\alpha, \tilde{\omega}^\alpha$, telles que :

1. l'exécution $(q_k, v_k^\alpha) \overset{\omega^\alpha}{\rightsquigarrow} (q_k, v_k^0)$ fait évoluer S α fois dans le cycle $\mathcal{C}_i$,
2. l'exécution $(\tilde{q}_k, v_k^\alpha) \overset{\tilde{\omega}^\alpha}{\rightsquigarrow} (\tilde{q}_{k'}, v_k^0)$ fait évoluer S dans le cycle $\overline{\mathcal{C}_i}$,
3. $P(\omega^\alpha) = P(\tilde{\omega}^\alpha)$.

D'après le lemme 3, il existe une paire de traces $\omega_k \in L_i$ et $\tilde{\omega}_k \in L - L_i$, telle que $(q_0, v_0) \overset{\omega_k}{\rightsquigarrow} (q_k, v_k^\alpha)$, $(q_0, v_0) \overset{\tilde{\omega}_k}{\rightsquigarrow} (\tilde{q}_k, v_k^\alpha)$ et $P(\omega) = P(\tilde{\omega})$.

Nous notons par ω et $\tilde{\omega}$, la concaténation de respectivement, ω_k et ω^α, et $\tilde{\omega}_k$ et $\tilde{\omega}^\alpha$; i.e., $\omega =$"$\omega_k.\omega^\alpha$" et $\tilde{\omega} =$"$\tilde{\omega}_k.\tilde{\omega}^\alpha$". Il est simple d'établir que $\omega \in L_i$ et $\tilde{\omega} \in L - L_i$ (puisque le cycle $\overline{\mathcal{C}_i}$ ne contient pas des transitions sur des défauts de l'ensemble F_i, sinon, l'étiquette $\mathcal{F}_i$ sera propagée dans tous les sommets du cycle $\overline{\mathcal{C}_i}$). Puisque $\Delta \leq \alpha.\delta$ et $\omega \in L_i^{\alpha.\delta}$, alors, $\omega \in L_i^{\Delta}$. En considérant que $P(\omega) = P(\tilde{\omega})$, nous concluons la non diagnosticabilité du langage L.

Preuve de la sufficance :
Considérons un diagnostiqueur D ne contenant ni un cycle F_i-indéterminé ni un sommet final F_i-incertain. Considérons $\omega \in L_i$. D'après le lemme 1, l'automate du diagnostiqueur D accepte le langage temporisé $P(L)$, en effet, il accepte la trace temporisée $P(\omega)$. Soit q^d un sommet du diagnostiqueur, accessible par une exécution de D sur la trace $P(\omega)$. Nous distinguons deux cas :

1. q^d est F_i-certain :
   D'après le lemme 4, $\forall \tilde{\omega} \in L \mid P(\omega) = P(\tilde{\omega}) \Rightarrow \tilde{\omega} \in L_i$. Nous concluons que L is diagnosticable.
2. q^d est F_i-incertain :
   Puisqu'il n'existe aucun cycle F_i-indéterminé et que q_d ne peut être final, deux cas sont possibles :
   (a) Il n'existe aucun cycle de sommets F_i-incertain dans le diagnostiqueur D. En plus, le diagnostiqueur séjourne dans chaque sommet F_i-incertain pour seulement une durée bornée de temps vue qu'aucun sommet F_i-incertain ne peut être aussi final. Puisque le diagnostiqueur comporte un nombre fini de sommets, une continuation assez longue de la trace ω ferait évoluer le diagnostieur vers un sommet F_i-certain, après une durée finie de temps (sinon, il évoluerait toujours dans des cycle F_i-incertain, ce qui contradit notre hypothèse sur l'absence de cycle de sommets F_i-incertain). Nous retrouvons ici le cas 2, et par conséquent, L est diagnosticable.
   (b) Il existe un ou plusieurs cycles de sommets F_i-incertain dans D, tels que aucun cycle ne lui correspond les cycles $\mathcal{C}_i$ et $\overline{\mathcal{C}_i}$ dans S. Nous réutilisons ici une partie de la preuve établie dans (Sampath et al., 1995). En effet, nous prouvons qu'il

ne serait pas possible de boucler pour un temps arbitrairement large dans ce cycle de sommets F_i-incertain, pour n'importe quelle longue continuation de ω, et par conséquent, un sommet F_i-certain sera atteint au bout d'un temps fini.

Considérons un élément $(q, \phi) \in Dis(l^d)$, tel que $\phi \neq \mathcal{F}_i$. Cet élément provient d'un élément $(\tilde{q}, \tilde{\phi}) \in Dis(\tilde{q}^d)$, où $\tilde{q}^d$ est le prédécesseur du sommet q^d dans le cycle et $\tilde{\phi} \neq \mathcal{F}_i$ (puisque les étiquettes de diagnostic sont propagées d'un sommet à son successeur).

En utilisant une induction en arrière, nous pouvons construire un cycle F_i-sûr dans S, contenant une partie, ou la totalité, des éléments des sommets F_i-incertain constituant le cycle. D'après notre hypothèse ; i.e., D ne contient aucun cycle F_i-indéterminé, ce cycle F_i-sûr n'a pas de cycle compagnon F_i-défaillant dans S. Par conséquent, une assez longue continuation de ω, fait évoluer S dans un chemin correspondant aux sommets contenant des éléments associés à l'étiquette $\mathcal{F}_i$ dans le diagnostiqueur. Ensuite, cette l'exécution sur cette trace va quitter le cycle de sommets F_i-incertain. Puisque cela est vrai pour tous les cycles F_i-incertain dans D, nous concluons que le diagnostiqueur quitte le cycle de sommets F_i-incertain au bout d'une durée finie de temps et évoluera ainsi vers un sommet F_i-certain. Nous retrouvons le cas 2 et par conséquent, L est diagnosticable.

4.6 Quelques éléments sur l'implémentation et la complexité de notre approche de diagnostic

Nous présentons dans cette section un aperçu sur quelques outils développés dans la littérature, permettant l'implémentation de notre algorithme de construction du diagnostiqueur. Ensuite, nous présentons une étude de la complexité des algorithmes employés par notre approche de diagnostic.

4.6.1 Quelques éléments sur l'implémentation de notre approche

Plusieurs outils ont été développés dans la littérature permettant la modélisation et l'analyse des automates temporisés. Nous pouvons citer l'outil de vérification des systèmes temporisés Uppaal (Larsen et al., 1997) qui correspond à l'un des outils les plus répandus dans le milieu industriel. Cet outil se base sur une représentation des zones d'horloges par une structure de données appelée DBM (signifie "Difference Bound Matrice"). Cette structure de donnée a été initialement développée pour la représentation des contraintes de différences (Cormen et al., 1990), ensuite, elle a été utilisée pour

l'analyse des automates temporisés (Alur, 1999). Les opérations nécessaires à l'analyse en avant ont été développées sur la base de DBM (Bengtsson et Yi., 2004). Afin d'avoir une représentation plus compacte des zones non-convexes, plusieurs améliorations ont été introduites sur cette structure de donnée telle que l'utilisation des CDD "Clock Difference Diagrams" (Pearson et al., 1998). Cette structure de donnée peut être envisagée pour l'implémentation de notre algorithme de synthèse du diagnostiqueur.

4.6.2 Etude de la décidabilité et de la complexité des algorithmes utilisés

Nous nous intéressons d'abord à l'étude de la décidabilité de l'algorithme de synthèse du diagnostiqueur. Ensuite, nous estimons la complexité de cet algorithme.

Nous avons souligné que l'analyse en avant de l'espace d'état d'un automate temporisé vérifiant l'hypothèse $\mathcal{H}_4$ est décidable. Cette hypothèse implique l'existence d'un nombre fini d'états symboliques accessibles. En effet, l'espace d'horloges accessible par le système correspond à un hypercube de dimension K, lorsqu'il s'agit de sommets non-finaux. Par ailleurs, l'ensemble d'états symboliques accessibles, associés à des sommets finaux, est fini puisque ces états n'admettent pas des successeurs discrets. L'arrêt de notre algorithme de construction du diagnostiqueur découle de la finitude de l'ensemble d'états symboliques accessibles. Ceci est justifié par le fait qu'un sommet du diagnostiqueur est composé d'un ensemble de couples (sommet,étiquette) associé à une zone d'horloge de l'espace d'état accessible. Puisque les ensembles correspondant à ces éléments sont finis, nous pouvons conclure que l'ensemble des sommets du diagnostiqueur est aussi fini, et par conséquent, l'algorithme de construction du diagnostiqueur s'arrête au bout d'un nombre fini de calcul de successeurs.

Théoriquement, la complexité de l'algorithme de construction du diagnostiqueur est doublement exponentielle en fonction de la taille de l'automate temporisé du système et de la constante choisie K. Cette grande complexité est attendue puisque nous manipulons des zones d'horloges non-convexes. Cependant, cette complexité est beaucoup moins importante dans la pratique. En effet, il est reconnu que l'utilisation d'algorithmes manipulant les zones d'horloges est plus efficace que ceux manipulant une abstraction plus fine telle que les régions (Alur, 1999). Cependant, la complexité théorique des algorithmes manipulant des régions est moins importante que ceux manipulant des zones.

La complexité de vérification de la diagnosticabilité est polynomiale en fonction du nombre des sommets du diagnostiqueur, puisqu'il s'agit d'un algorithme de détection de cycles.

4.7 Conclusion

Nous avons présenté au cours de ce chapitre le principe de notre approche de diagnostic de SED à base d'automates temporisés.

Dans un premier volet, nous avons défini le cadre de modélisation de notre approche. Ainsi, le système à diagnostiquer est modélisé par un automate temporisé décrivant à la fois son comportement normal et ses comportements défaillants. Ce modèle doit respecter certaines spécifications et hypothèses indispensables pour la construction du diagnostiqueur. Ensuite, nous avons présenté, intuitivement puis formellement, notre approche de diagnostic qui se base sur la construction d'un diagnostiqueur. Ce diagnostiqueur est un automate temporisé déterministe qui évolue en fonction des événements observables générés par le système et estime l'état courant du système et les occurrences de défauts. Un fonction de décision, analyse le sommet courant du diagnostiqueur et annonce l'occurrence d'un défaut du mode F_i lorsque ce sommet est F_i-certain.

Dans un deuxième volet, nous avons étudié la diagnosticabilité du système à diagnostiquer. Nous avons établi que l'utilisation d'un diagnostiqueur est soumise à la vérification de notion de diagnosticabilité par le modèle du système considéré. Ensuite, nous avons présenté une méthode systématique permettant de vérifier la diagnosticabilité d'un modèle. Cette méthode repose sur la détection dans le modèle du diagnostiqueur des cycles F_i-indéterminé et des sommets finaux F_i-incertain. Un théorème liant la notion de diagnosticabilité et la vérification de ces conditions a été formalisé. Enfin, nous avons présenté une étude préliminaire de la complexité des algorithme présenté ainsi que quelques détails liés à l'implémentation de notre approche.

Dans le chapitre suivant, nous allons proposer une approche de diagnostic pour les SDH. L'utilisation des modèles hybrides pour le diagnostic des SDH représente un grand défit à remonter vue les nombreux problèmes indécidables liées à ses modèles et leur grande complexité. Ainsi, peu de concepts, développés dans notre démarche de diagnostic à base d'automates temporisés, peuvent être appliqués dans le cas de modèles hybrides.

Chapitre 5

Diagnostic des systèmes hybrides rectangulaires

5.1 Introduction

Les systèmes de production complexes peuvent être vus comme des systèmes dynamiques hybrides ayant des dynamiques continues et divers aspects discrets. L'utilisation d'une approche de diagnostic reposant sur une abstraction des dynamiques continues d'un SDH, à travers l'utilisation de modèles SED, entraîne une perte considérable d'informations, parfois indispensables pour l'identification des défauts. Ainsi, dans certains cas, les comportements défaillants se manifestent par une déviation des trajectoires des dynamiques continues du système. Ceci rend l'utilisation d'une démarche de diagnostic fondée sur une abstraction purement discrète de l'évolution du système, inadéquate pour l'identification des défauts.

Afin de pallier à ce problème, nous avons proposé dans le chapitre précédent, une démarche de diagnostic qui prend en considération l'aspect temporel dans l'évolution du SED à diagnostiquer. Cette approche repose sur l'utilisation d'un modèle complet du système sous la forme d'un automate temporisé. Ce modèle décrit l'évolution temporelle des événements du système à travers l'utilisation d'un ensemble d'horloges. Ces variables fictives permettent **d'abstraire les intéractions** entre les **dynamiques continues** (les variables) et les **dynamiques discrètes** (les événements) d'un système, en utilisant un ensemble de contraintes temporelles conditionnant les évolutions discrètes. Par exemple, nous supposons qu'un capteur de niveau génère l'événement *plein* lorsque le niveau du liquide dans un bac, qu'on modélise avec une variable x, atteint une valeur seuil $x = 200$, sachant que le remplissage s'effectue avec une dynamique $\dot{x} \in [4, 5]$ à partir du niveau initial $x = 0$. Cette évolution du système peut être modélisé à travers l'utilisation d'une horloge fictive y qui mesure le temps écoulé à partir du début de la phase de remplissage,

et en associant la contrainte $y \in [40, 50]$ à la transition sur l'événement *plein*. Dans cet exemple, l'utilisation de l'horloge y permet d'abstraire la variable continue x.

Une telle abstraction des dynamiques continues à travers l'utilisation de contraintes sur des horloges n'est pas simple à réaliser surtout lorsqu'il s'agit d'un système complexe comportant plusieurs variables continues. Dans certains cas, cette abstraction n'est même pas possible puisque les modèles hybrides sont strictement plus expressifs que les modèles SED temporisés (Henzinger et al., 1998).

Ainsi, il serait plus convenable d'utiliser un modèle concis des systèmes réels qui décrit intrinsèquement l'interaction entre son aspect continu et son aspect discret, tel que le modèle automate hybride. Le modèle automate hybride représente un outil puissant permettant la modélisation et l'analyse des SDH. Cet outil bénéficie d'une grande flexibilité et d'une puissante expressivité (Henzinger, 1996). Toutefois, l'utilisation de ce modèle est confrontée à l'indécidabilité de certains problèmes fondamentaux tel que l'analyse d'accessibilité et le problème de déterminisation.

C'est dans ce cadre que nous présentons notre démarche de diagnostic pour une sous-classe de SDH. Notre approche repose sur la modélisation du comportement complet du système à travers une sous-classe simple d'automates hybrides : les automates hybrides rectangulaires, sous certaines conditions. Ce formalisme permet d'approximer les comportements de la dynamique continue du système à travers l'utilisation de conditions de flux rectangulaires de la forme $\dot{x} \in [a, b]$. Dans ce chapitre, nous introduisons d'abord le modèle automate hybride rectangulaire. Ensuite, nous présentons une procédure de diagnostic qui représente le cœur de notre démarche de diagnostic pour les SDH. Nous définissons une notion de diagnosticabilité, dite diagnosticabilité à horizon de temps limité, pour les langages temporisés acceptés par les automates rectangulaires hybrides considérés dans notre contribution. Enfin, une méthode systématique permettant la vérification de cette notion est décrite.

5.2 Les Automates Hybrides Rectangulaires

Nous présentons dans cette partie le modèle Automate Hybride Rectangulaire (AHR), qui constitue le cadre de modélisation de l'approche de diagnostic que nous désirons développer dans ce chapitre. Ce modèle peut être considéré comme une généralisation du modèle automate temporisé, où l'évolution continue n'est plus représentée par des horloges mais par des équations différentielles sur les variables continues du système.

Dans la suite de cette section, nous présentons formellement le modèle automate hybride rectangulaire, en rappelant les aspects syntaxique et sémantique de ce modèle. Ensuite, nous introduisons une méthode permettant l'analyse comportemenentale d'un SDH modélisé par un AHR. Cette méthode repose sur l'application d'une fonction d'ana-

lyse d'accessibilité en avant sur le modèle considéré.

5.2.1 Syntaxe

Soit $X = \{x_1, \ldots, x_n\}$ un ensemble de variables réelles. Nous notons par $\dot{X} = \{\dot{x} \mid x \in X\}$ l'ensemble des dérivées premières des variables dans X par rapport au temps. Une variable x est dite un *chronomètre*, si $\dot{x} \in \{0, 1\}$, et dite une horloge, si $\dot{x} = 1$.

Définition 30.

- Une *inégalité rectangulaire* sur X est une inégalité de la forme $x \sim c$, où $x \in X$, $c \in \mathbb{Z}$ et $\sim \in \{<, \leq, =, \geq, >\}$.
- Un *prédicat rectangulaire* sur X est une conjonction d'inégalités rectangulaires sur X. Nous notons par $Rect(X)$ l'ensemble des prédicats rectangulaires sur X.
- Une *inégalité polyhédrale* sur X est une inégalité de la forme $c_1x_1 + \cdots + c_kx_k \sim c$, où $x_1, \ldots, x_k \in X$ et $c, c_1, \ldots, c_k \in \mathbb{Z}$.
- Un *prédicat polyhédral* (ou un *polyhèdre*) sur X est une conjonction d'inégalités polyhédrales sur X. Nous désignons par $\Psi(X)$ l'ensemble de prédicats polyhédraux X.

Définition 31. Une valuation sur X est un vecteur de $\mathbb{R}^n$, $\mathbf{v} = (v_1, \ldots, v_n)$, qui définit une valeur $v_i \in \mathbb{R}$ pour chaque variable $x_i \in X$.

Définition 32. Une *region* sur l'ensemble X est un sous-ensemble de $\mathbb{R}^n$. Pour chaque région z et $x_i \in X$, $z(x_i) = \{v_i \in \mathbb{R} \mid \mathbf{v} \in z\}$. Nous notons par $[\![\psi]\!]$ la région composée par l'ensemble des vecteurs $\mathbf{v} \in \mathbb{R}^n$, où le prédicat ψ est vrai si on remplace x_i par la valeur correspondante v_i, pour chaque $i \in \{1, \ldots, n\}$.

En l'absence de toute ambiguïté, nous pouvons désigner une région $[\![\psi]\!]$ par l'expression du prédicat correspondant ψ. Pour un prédicat rectangulaire ψ et une variable x_i, nous désignons par $[\![\psi]\!](x_i)$ l'intervalle de valeurs décrit par les $i^{ème}$ composants des vecteurs vérifiant $\mathbf{v} \in [\![\psi]\!]$.

Définition 33. Un *Automate Hybride Rectangulaire* (Henzinger et al., 1998) est un septuplet $H = (Q, X, \Sigma, E, inv, flux, init, M)$, où :

- $Q = \{q_1, \ldots, q_k\}$ est un ensemble fini de sommets représentant les états discrets du système ;
- X est un ensemble fini de variables réelles. L'état continu du système est caractérisé à tout instant par un vecteur $\mathbf{v} = (v_1, \ldots, v_n)$ dans l'espace Euclidien $\mathbb{R}^n$, où v_i correspond à la valeur de la variable x_i ;
- Σ est un ensemble d'événements ;

- $E \subseteq Q \times \Sigma \times Rect(X) \times Rect(X) \times 2^X \times Q$ est un ensemble fini de transitions. Une transition (q, σ, g, r, R, q') correspond à un changement de sommet de q à q', sur l'occurrence de l'événement σ et sous la condition $\mathbf{v} \in [\![g]\!]$, où le vecteur $\mathbf{v}$ correspond aux valeurs courantes des variables de X. Après le franchissement de la transition, chaque variable $x_i \in R$ est réinitialisée, d'une manière non déterministe, par une valeur dans l'intervalle $[\![r]\!](x)$. Les autres variables qui ne sont pas dans R, restent inchangées. Nous utilisons l'opérateur $Source(e)$, $e \in E$, pour désigner le sommet source de la transition e ;
- $inv : Q \rightarrow Rect(X)$ est une fonction qui associe à chaque sommet $q \in Q$ une contrainte rectangulaire pour chaque variable $x_i \in X$. Le système peut séjourner dans un sommet tant que l'invariant du sommet est satisfait ; i.e., la valeur de chaque variable $x_i \in X$ est incluse dans l'intervalle $[\![inv(q)]\!](x_i)$;
- $flux : Q \rightarrow Rect(\dot{X})$ est la fonction qui affecte à chaque sommet une représentation pour l'évolution continue. Durant le séjour dans un sommet q de l'automate, l'évolution des variables continues est exprimée généralement sous la forme d'une équation d'état $flux(q)$, où pour chaque variable $x_i \in X$, sa première dérivée par rapport au temps $\dot{x}_i$ doit évoluer dans l'intervalle $[\![flux(q)]\!](\dot{x}_i)$;
- $init \subseteq Q \times Rect(X)$, désigne la condition initiale de l'automate ;
- $M \subseteq Q$ correspond à l'ensemble des sommets marqués de l'automate.

Définition 34. Un AHR H admet un *non-déterminisme borné* (Henzinger et al., 1998), si :

1. les régions associées à $init$ et $flux(q)$ sont bornées pour chaque $q \in Q$;
2. pour chaque transition $(q, \sigma, g, r, R, q') \in E$ et $x_i \in R$, l'intervalle $[\![r]\!](x_i)$ est borné.

Dans la suite de notre travail, nous considérons uniquement les AHRs ayant un non-déterminisme borné. Par ailleurs, nous supposons que pour chaque sommet q et chaque variable $x_i \in X$, les bornes de l'intervalle $[\![flux(q)]\!](\dot{x}_i)$ doivent être des valeurs entières, en plus, $[\![flux(q)]\!](\dot{x}_i) \subset (-\infty, 0]$ ou $[\![flux(q)]\!](\dot{x}_i) \subset [0, +\infty)$.

Définition 35. Un *Automate Hybride Rectangulaire Initialisé* est un AHR où pour chaque transition, les variables qui changent de contraintes de flux lors du franchissement de la transition seront réinitialisées. Formellement, pour chaque transition $(q, \sigma, g, r, R, q') \in E$ et chaque variable $x \in X$, si $[\![flux(q)]\!](\dot{x}) \neq [\![flux(q')]\!](\dot{x})$ alors $x \in R$.

Exemple 5.2.1
La figure 5.1 illustre l'exemple d'un automate hybride rectangulaire initialisé. Nous pouvons constater que ce modèle comporte trois états discrets et deux variables x_1 et x_2. Par ailleurs, chacune de ces variables est réinitialisée lorsqu'elle change de flux entre deux sommets. La condition initiale de ce sommet correspond à la région $\langle x_1 = 0 \wedge x_2 = 0 \rangle$.

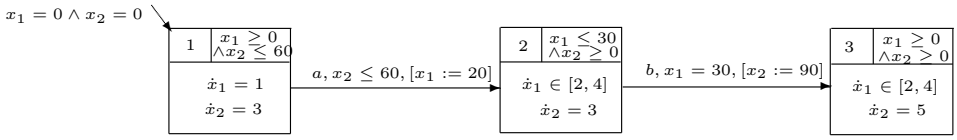

Fig. 5.1 – *Exemple d'un automate hybride rectangulaire initialisé.*

Définition 36. Soient $H_1 = (Q_1, X_1, \Sigma_1, E_1, inv_1, flux_1, init_1, M_1)$ et $H_2 = (Q_2, X_2, \Sigma_2, E_2, inv_2, flux_2, init_2, M_2)$ deux AHRs, tel que $Q_1 \cap Q_2 = \{\}$. Le produit synchrone de H_1 et H_2, noté par $H_1 \otimes H_2$, correspond à l'AHR $H = (Q, X, \Sigma_1 \cup \Sigma_2, E, inv, flux, init, M)$, où $Q = Q_1 \times Q_2$, $X = X_1 \cup X_2$, $init = init_1 \wedge init_2$ et pour chaque paire de sommets $q_1 \in Q_1, q_2 \in Q_2$, $flux((q_1, q_2)) = flux(q_1) \wedge flux(q_2)$ et $inv((q_1, q_2)) = inv_1(q_1) \wedge inv_2(q_2)$.

Une transition $e = ((q_1, q_2), \sigma, g, r, R, (q'_1, q'_2)) \in E$ correspond à l'une des situations suivantes :

1. $\sigma \in \Sigma_1 \backslash \Sigma_2$, $(q_1, \sigma, g, r, R, q'_1) \in E_1$ et $q_2 = q'_2$,
2. $\sigma \in \Sigma_2 \backslash \Sigma_1$, $(q_2, \sigma, g, r, R, q'_2) \in E_2$ et $q_1 = q'_1$,
3. $\sigma \in \Sigma_1 \cap \Sigma_2$, $(q_1, \sigma, g_1, r_1, R_1, q'_1) \in E_1$, $(q_2, \sigma, g_2, r_2, R_2, q'_2) \in E_2$, $g = g_1 \wedge g_2$, $r = r_1 \cup r_2$ et $R = R_1 \cup R_2$.

Un sommet (q_1, q_2) de H est marqué : $(q_1, q_2) \in M$, si $q_1 \in M_1$ ou $q_2 \in M_2$.

5.2.2 Sémantique

A chaque instant, l'état d'un automate hybride rectangulaire est donné par une paire $(q, \mathbf{v})$ correspondant à l'association d'un état discret du système q et d'un vecteur $\mathbf{v} \in \mathbb{R}^n$ indiquant la valeur courante de chaque variable.

Nous présentons la sémantique d'un automate hybride rectangulaire H, en terme de tous les comportements qui peuvent être générés par l'évolution du système modélisé. Dans ce sens, la sémantique d'un système continu, définie par un ensemble d'équations différentielles, est donnée par l'ensemble de toutes ses solutions, notamment de toutes ses trajectoires.

L'évolution d'un modèle automate rectangulaire hybride est réalisée selon deux types de transitions :

- transitions continues (figure 5.2-a) : la partie discrète de l'état reste constante dans le même sommet q, tandis que la partie continue évolue de la valuation $\mathbf{v}$ à la valuation $\mathbf{v}'$. Cette évolution se fait via une trajectoire continue, décrite par les dynamiques des variables évoluant dans des intervalles de flux spécifiés par $flux(q)$, tout en respectant les contraintes d'invariants décrites par $inv(q)$. Cette

transition est notée par $(q, \mathbf{v}) \xrightarrow{\delta} (q, \mathbf{v}')$, où $\delta \in \mathbb{R}_+$ désigne le temps écoulé durant la transition ;

- transitions discrètes (figure 5.2-b) : correspondent aux changements discrets et instantanés de sommets comme décrit par le quintuplet $e = (q, \sigma, g, r, R, q') \in E$. Cette transition est notée par $(q, \mathbf{v}) \xrightarrow{e} (q', \mathbf{v}')$. Une transition ne peut être franchie que si le prédicat de la garde est satisfait par la partie continue de l'état : $\mathbf{v} \in [\![g]\!]$. Dans ce cas, la partie discrète change du sommet q au sommet q' et la partie continue est mise à jour selon l'ensemble des affectations de variables.

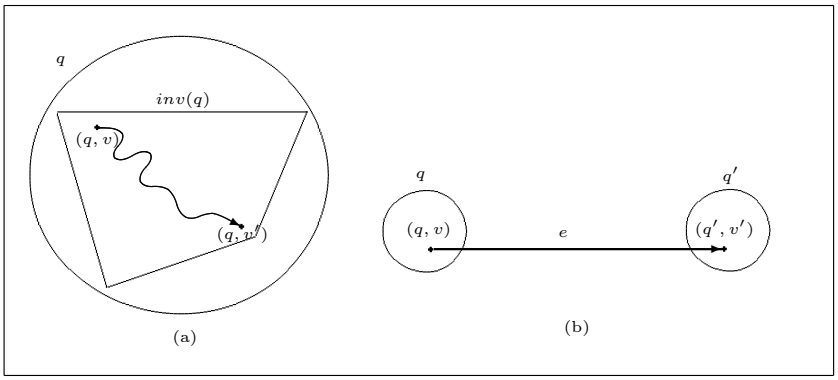

Fig. 5.2 – *Transitions continues et transitions discrètes d'un AHR.*

Ainsi, le système passe du temps dans chaque état discret, permettant ainsi aux variables continues d'évoluer et effectue des transitions discrètes entre ces états. La dynamique associée aux variables continues est spécifiée par la fonction de flux. Le franchissement d'une transition discrète est synchronisé avec l'occurrence d'un événement. La valuation des variables dans X lors de ce franchissement doit vérifier la condition de garde définie. Une transition est dite *autonome*, si elle est franchie dès que sa condition de garde est vérifiée. Dans notre travail, la syntaxe définissant un AHR ne considère pas les transitions autonomes. Par contre, nous pouvons avoir des transitions équivalentes en définissant des transitions sur des événements non observables et des conditions des gardes et d'invariants bien déterminées.

Définition 37.

- Une *exécution* d'un AHR H est une séquence finie ou infinie de transitions continues et/ou discrètes $(q_0, \mathbf{v_0}) \rightarrow (q_1, \mathbf{v_1}) \rightarrow (q_2, \mathbf{v_2}) \rightarrow \ldots$, où $(q_0, \mathbf{v_0}) \in init$.
- Un état $(q, \mathbf{v})$ est dit *accessible* (ou atteignable), s'il existe une exécution de H qui rejoint cet état à partir d'un état initial $(q_0, \mathbf{v}_0) \in init$.

Dans la suite, nous rappelons quelques définitions, introduites au cours du chapitre précédent. Une trace temporisée est définie comme une séquence, finie ou infinie, d'événements alternés avec des réels positifs : $\delta_0, \sigma_1, \delta_1, \sigma_2, \delta_2, \ldots, \sigma_k, \delta_k \ldots$, où les σ_i, pour $i \geq 1$,

sont des éléments de Σ et $\delta_i \in \mathbb{R}_+$, correspond au temps écoulé entre les occurrences des événements successifs σ_i et σ_{i+1}.

Une trace temporisée ω est dite *acceptée* par un AHR H, s'il existe une exécution de H étiquetée avec les éléments de ω. L'opérateur $temps(\omega)$ renvoie la durée d'une trace temporisée ω.

Définition 38. Un AHR H est dit *Fortement Non-Zénon* (FNZ) (Roux et Rusu, 1996), s'il existe un entier naturel $d > 0$, pour lequel chaque exécution inscrite dans un cycle de transitions de H admet une durée supérieure à d. Nous pouvons facilement établir que les conditions suivantes garantissent le fait qu'un automate temporisé H est fortement non-zénon :

pour chaque cycle de transitions dans H, $q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{k-1}} q_k \xrightarrow{e_k} q_1$:

- il existe au moins une variable $x \in X$, telle que $\dot{x}$ admet une borne inférieure strictement positive dans tous les sommets du cycle, ou $\dot{x}$ admet une borne supérieure strictement négative dans tous les sommets du cycle ;
- il existe deux transitions e, e' dans ce cycle, où x n'est pas réinitialisée dans les transitions entre e et e', et deux entiers $c, c' \in \mathbb{Z}$, avec $c < c'$, tels que :
  - si $\dot{x}$ est strictement positive alors la variable x est réinitialisée à c dans e et admet comme borne inférieure c' ; i.e., $x \geq c'$, dans la condition de garde de e'.
  - si $\dot{x}$ est strictement négative alors x est réinitialisée à c' dans e et admet comme borne supérieure c ; i.e., $x \leq c$, dans la condition de garde de e'.

Dans un AHR fortement non-zénon, les traces infinies sont nécessairement à temps-divergent comme il est le cas pour les automates temporisés. En effet, après le parcours de chaque cycle de transitions, le temps d'une exécution de l'AHR progresse uniformément avec une durée minimale, supérieure à d u.t..

Exemple 5.2.2
La figure 5.3 présente un exemple d'automate hybride rectangulaire fortement non-zénon. Nous pouvons constater que la variable x_1 garde une dynamique strictement positive dans tous les états discrets de l'automate. En plus cette variable est réinitialisée à 0, dans la transition entre les sommets 3 et 1, et admet comme borne inférieure la valeur 30 dans la contrainte de garde de la transition entre les sommets 2 et 3. Ainsi, dans chaque exécution de l'automate, parcourant ce cycle de sommets, s'écoule une durée de temps supérieure à 15 u.t..

Proposition 1. Soient H_1 and H_2 une paire de AHRs et soit $H = H_1 \otimes H_2$ leur produit synchrone, alors :

1. si H_1 et H_2 sont initialisés alors H est initialisé,
2. si H_1 et H_2 sont fortement non-zénon alors H est fortement non-zénon.

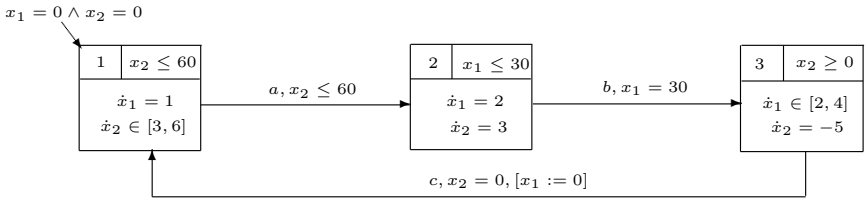

Fig. 5.3 – *Exemple d'un automate hybride rectangulaire fortement non-zénon*

Preuve :

1) Immédiate d'après la définition d'un AHR initialisé.

2) Chaque cycle de transitions dans l'AHR H correspond à un cycle dans H_1 ou H_2, ou la synchronisation de deux cycles, le premier dans H_1 et le second dans H_2. Par conséquent, dans chaque exécution inscrite dans un cycle de H s'écoule une durée de temps supérieure à $d > 0$, qui correspond à la durée d'exécution d'un cycle dans H_1 et/ou dans H_2.

5.2.3 Analyse d'accessibilité d'un AHR

L'analyse d'accessibilité constitue un problème central dans la vérification des propriétés des systèmes hybrides modélisés par des automates hybrides rectangulaires. Cette analyse est généralement basée sur le calcul d'un ensemble d'états accessibles à partir d'un ensemble d'états donné. En effet, nous cherchons à vérifier si une région de l'espace d'état est accessible à partir d'une région initiale donnée.

Comme il est le cas pour les automates temporisés, l'espace d'état d'un AHR est infini et non dénombrable. Ainsi, pour mener le calcul d'accessibilité il faut disposer d'un formalisme de représentation permettant de manipuler les éléments de cet espace d'état. Dans notre travail, nous avons choisi une représentation basée sur la notion de région d'état. Une *région* ou (un *état symbolique*) d'un AHR est représentée par une paire (q, z), où q correspond à un sommet d'automate et z une région de l'espace d'état continu, représentée par un polyhèdre. Un état $(q', \mathbf{v})$ est inclus dans une région (q, z) si $q = q'$ et $\mathbf{v} \in z$. En effet, un état symbolique (q, z) permet de représenter l'ensemble d'états $\{(q, \mathbf{v}) \mid \mathbf{v} \in z\}$.

L'analyse d'accessibilité pose le problème de l'existence d'une trajectoire à partir d'une région initiale (q_0, z_0) qui peut amener l'état du système dans une région finale (ou marquée) (q_m, z_m). Dans la littérature, on utilise généralement la méthode d'analyse en avant pour vérifier l'accessibilité d'une région de l'espace d'état. Il s'agit de calculer, d'une manière itérative, l'ensemble de tous les successeurs des états d'une région initiale (q_0, z_0). L'ensemble des états cibles (les états marqués qu'on désire atteindre) est donné

par (q_m, z_m).

Les successeurs d'un état symbolique sont calculés itérativement, en alternant le calcul de successeurs continus et discrets (Roux et Rusu, 1996) (figure 5.4).

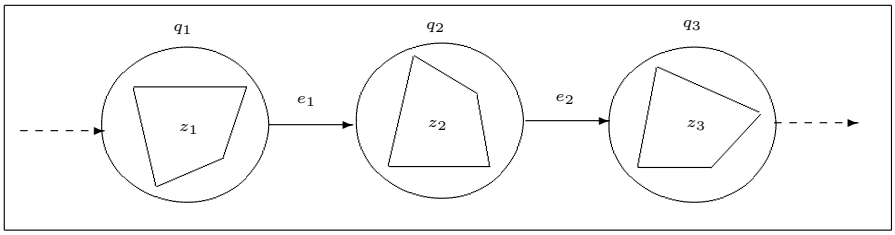

Fig. 5.4 – *Analyse d'accessibilité symbolique d'un AHR.*

Le calcul du successeur continu d'un état symbolique (q, z) est réalisé en utilisant l'opérateur $Post_c$, défini de la manière suivante :

$$Post_c((q,z)) = \{(q,\mathbf{v}')|(q,\mathbf{v}) \xrightarrow{\delta} (q,\mathbf{v}'), \mathbf{v} \in z, \delta \in \mathbb{R}_+\}.$$

De même, l'opérateur successeur discret d'une région (q, z) sur le franchissement d'une transition $e \in E$, noté $Post_d((q,z),e)$, est défini de la manière suivante :

$$Post_d((q,z),e) = \{(q',\mathbf{v}')|(q,\mathbf{v}) \xrightarrow{e} (q',\mathbf{v}'), \mathbf{v} \in z\}.$$

On définit l'opérateur $Post$ qui correspond à la composition de l'opérateur successeur discret, sur le franchissement d'une transition $e \in E$, et l'opérateur successeur continu :

$$Post((q,z),e) = Post_c \circ Post_d((q,z),e).$$

Pratiquement, le calcul d'une région (q', z'), successeur d'une région (q, z) sur une transition $e = (q, \sigma, g, r, R, q')$, est effectué en appliquant les étapes suivantes (Guéguen et Zaytoon, 2004) :

1. calculer z_1 qui correspond à l'intersection de l'invariant du sommet q ; i.e., $inv(q)$, et le futur de la région z. Le futur d'une région correspond à l'ensemble des états accessibles à partir de la région de départ en appliquant la fonction de flux (voir l'exemple dans la figure 5.5) ;
2. calculer z_2 qui correspond à l'intersection de z_1 avec la condition de garde g ;
3. calculer z_3 en appliquant la fonction de réinitialisation définie par R et r sur la région z_2 ;
4. calculer z_4 qui correspond à l'intersection de la région z_3 et l'invariant du sommet q' ; i.e., $inv(q')$;

5. enfin, calculer z' qui correspond à l'intersection de l'invariant du sommet q' ; i.e., $inv(q')$, et le futur de la région z_4.

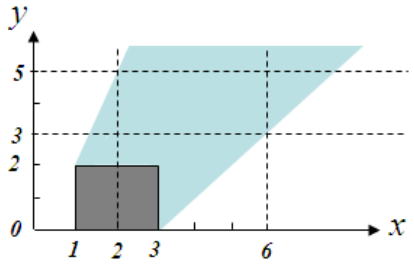

Fig. 5.5 – Futur d'un polyhèdre $\langle x \in [1,3] \wedge y \in [0,2] \rangle$ sur une fonction de flux $\langle \dot{x} = 1 \wedge \dot{y} \in [1,3] \rangle$

Dans la suite, nous définissons la fonction d'analyse d'accessibilité, permettant de déterminer l'existence d'un état marqué accessible par une exécution d'un AHR ; autrement dit, l'existence d'une région $(q_m, \langle vrai \rangle)$, $q_m \in M$, accessible à partir de la région (q_0, z_0). Ainsi, une fois qu'un état marqué (sa partie discrète $q \in M$) est accessible, la fonction renvoie "OUI". Par contre, si aucun état marqué n'est accessible, elle renvoie "NON".

Algorithm 4 Fonction d'analyse d'accessibilité : *Accessible*

Entrées : AHR H
Sorties : {"OUI","NON"}

$A\_TRAITER \leftarrow \{(q_0, z_0)\}$; $RÉGIONS\_TRAITÉES \leftarrow \{\}$;
tantque $A\_TRAITER \neq \{\}$ **faire**
  retirer (q, z) de $A\_TRAITER$;
  si $q \in M$ **alors**
    retourner "OUI" ;
  sinon si $z \not\subseteq z_k, (\forall (q, z_k) \in RÉGIONS\_TRAITÉES)$ **alors**
    ajouter (q, z) à $RÉGIONS\_TRAITÉES$
    pour tout $e \in E$ tel que $q = Source(e)$ **faire**
      $(q', z') = Post((q, z), e)$;
      ajouter (q', z') à $A\_TRAITER$, si $z' \neq \emptyset$;
    fin pour
  finsi
fin tantque
retourner "NON" ;

Cet algorithme de calcul d'accessibilité peut ne pas converger lorsqu'il est appliqué sur un automate hybride linéaire sans restrictions. D'un point de vue théorique cette

non convergence s'explique par le fait que le problème d'analyse d'accessibilité d'un tel automate est non-décidable (Alur et al., 1995) ; i.e, il n'est pas possible de trouver un algorithme qui réalise ce calcul et dont la convergence est assurée en un temps fini.

Cependant, des cas particuliers (sous-classes d'automates hybrides linéaires) de convergence de cet algorithme ont été identifiés dans la littérature, il s'agit du modèle automate hybride rectangulaire initialisé. En effet, il a été prouvé dans (Henzinger et al., 1998) que l'analyse d'accessibilité pour ce modèle est décidable et que l'algorithme d'accessibilité en avant admet une complexité PSPACE.

5.3 Vers une approche de diagnostic pour les SDH

Les modèles des systèmes hybrides se distinguent par leur grande expressivité, leur capacité à représenter intrinsèquement les interactions entre les dynamiques continues et discrètes d'un système. Toutefois, le développement d'applications à partir de ces modèles, telles que la synthèse de superviseurs, de diagnostiqueur ou le model-checking peut se confronter aux résultats d'indécidabilités liés à la plupart des problèmes standards : vérification du vide, universalité, analyse d'accessibilité, déterminisation ..., (Alur et al., 1995). Ces indécidabilités nous conduisent à considérer des sous-classes de modèles hybrides, ayant des dynamiques plus simples.

En effet, dans le cadre de notre travail, nous utilisons une sous-classe des automates hybrides rectangulaires pour modéliser le système à diagnostiquer. Malgré la simplicité de la dynamique continue de ce modèle, il est doté d'une grande expressivité à travers sa capacité à représenter, d'une manière approximative, la dynamique discrète et la dynamique continue d'un SDH en utilisant des bornes inférieures et supérieures sur les dérivées des variables continues (par exemple, $\dot{x} \in [2, 5]$) (Kopke, 1996).

Toutefois, le problème de synthèse hors-ligne d'un diagnostiqueur à partir d'un modèle automate hybride rectangulaire est indécidable. Ce résultat est attendu puisque ce problème est aussi indécidable pour un cas particulier de AHR à savoir, le modèle automate temporisé (sans restrictions).

C'est à partir de ces constatations que nous proposons une solution de diagnostic en-ligne d'une sous-classe de SDH modélisée par des automates hybrides rectangulaires. Dans notre solution, nous nous sommes inspirés d'un ensemble de techniques développées dans les travaux de Tripakis (Tripakis, 2002) et Sampath (Sampath et al., 1995, 1996). Elle repose sur l'utilisation d'une *procédure de diagnostic* sous la forme d'un algorithme d'estimation d'état, exécuté en-ligne avec le système à diagnostiquer. La procédure de diagnostic est passive dans le sens où elle n'influence pas le fonctionnement du système à diagnostiquer.

Dans la figure 5.6, nous illustrons le schéma global de notre solution de diagnostic. Cette solution s'appuie sur un modèle "complet" du SDH à diagnostiquer, décrivant son comportement normal et ses possibles comportements anormaux (défaillants). Ce modèle est donné sous la forme d'un automate hybride rectangulaire, vérifiant certaines hypothèses. A partir de ce modèle, des événements observables générés par le système et les délais de temps séparant ces événements, la procédure de diagnostic estime l'état courant du système ainsi que les défauts non observables qui affectent son fonctionnement. Nous notons que la procédure n'a **aucune connaissance directe** sur les valeurs des **variables continues**. Ces valeurs seront estimées en analysant le comportement discret observable du système. Le diagnostic des défauts est ainsi élaboré **uniquement** à partir de l'analyse du **comportement discret observable** du système, en s'appuyant également sur **l'aspect temporel** qui caractérise cette évolution.

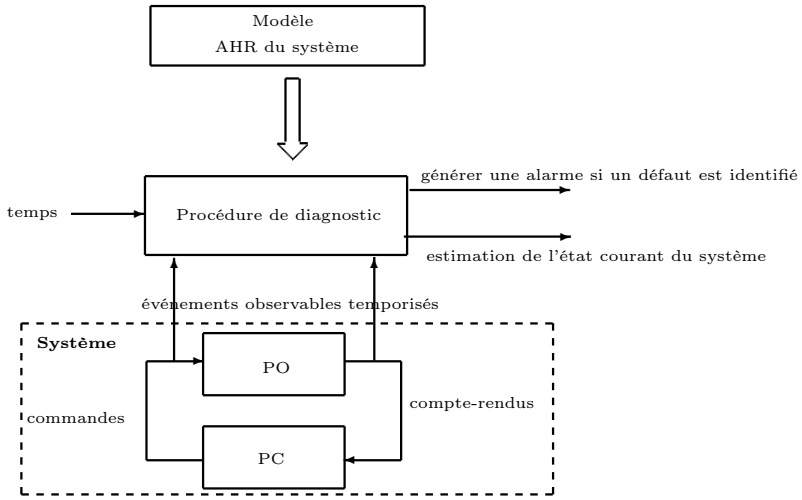

Fig. 5.6 – *Schéma du diagnostic en-ligne à base d'automates hybrides rectangulaires*

Afin de présenter le principe de notre approche de diagnostic pour les SDH, nous considérons, tout au long de ce chapitre, l'exemple du système de chauffage de liquides introduit dans le chapitre 4. Dans la suite, nous décrivons ce système en considérant l'évolution de sa dynamique continue.

Exemple 5.3.1

Le système de chauffage de liquides, illustré dans la figure 5.7(a), comporte deux vannes V_1 and V_2, un bac, un thermostat et deux capteurs de niveau : le capteur S_1 qui surveille le niveau maximal et le capteur S_2 qui surveille le niveau minimal. Le liquide à chauffer est introduit par la vanne V_1 avec un débit $\dot{x}_1 \in [3, 5]$. Quand le niveau de ce liquide, représenté par la variable x_1, atteint le niveau maximal $x_1 = 500$, le capteur S_1 génère un événement de notification au contrôleur pour qu'il commande la fermeture de la vanne V_1.

Par la suite, le liquide est chauffé pendant une durée de 40 u.t., mesurée par l'horloge x_2. Le liquide est ensuite évacué à travers la vanne V_2 avec un débit $\dot{x}_1 \in [-8, -6]$ jusqu'à ce que le bac soit vide ; i.e., $x_1 = 0$. A ce moment, le capteur S_2 génère un événement de notification, le contrôleur commence alors un nouveau cycle de chauffage.

La figure 5.7(b) illustre un modèle automate hybride rectangulaire décrivant le comportement normal et les comportements défaillants de ce système. Ce modèle ne représente que quelques aspects du fonctionnement de ce système. En effet, pour des raisons de clarté, certains événements, jugées non pertinents pour la compréhension de notre exemple, n'ont pas été illustrés (comme les événements d'ouverture et de fermeture des vannes). Nous supposons que le fonctionnement de ce système peut être affecté par deux défauts non observables :

- le premier défaut correspond à l'existence d'une fuite dans le bac. Ce défaut est représenté par l'événement $fuite$;
- le deuxième défaut correspond au blocage de la vanne V_1 en position fermée. Ce défaut est représenté par l'événement $blocage\_V_1$.

Les événements s_1 et s_2 sont observables et correspondent aux notifications générées par, respectivement, par les capteurs S_1 et S_2. L'événement u est non observable. Cet événement a été introduit pour permettre la représentation d'une transition autonome, franchie lorsque l'horloge x_2 atteint la valeur 40. Le système admet un ensemble d'états initiaux désigné par la région $(1, \langle x_1 \in [0, 500] \wedge x_2 = 0 \rangle)$.

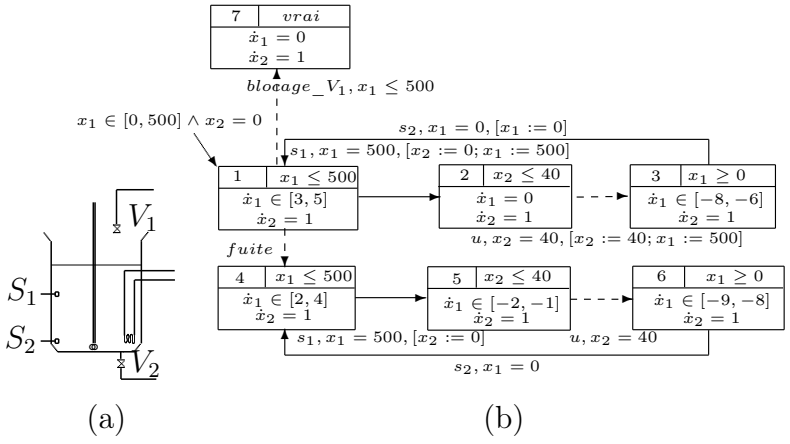

Fig. 5.7 – *Modèle automate hybride rectangulaire d'un système de chauffage de liquides*

En se focalisant sur les différentes valeurs prises par la dérivée de la variable x_1, nous pouvons constater que l'occurrence d'une fuite affecte les débits d'entrée/sortie du liquide durant les phases de remplissage, de chauffage et d'évacuation. Ce changement dans la dynamique de x_1 peut affecter les dates d'occurrence des événements observables

qui suivent ce défaut, ce qui constitue la signature temporelle du défaut. La procédure de diagnostic permet d'exploitation de cette signature afin d'identifier le défaut correspondant. Dans cet exemple, la procédure estime les différentes valeurs possibles de la variable x_1 en se basant sur la durée de temps qui sépare les occurrences de s_1 et s_2. En effet, si ce délai est trop court, les valeurs estimées pour x_1 correspondent à des valeurs strictement supérieures à 0. Par conséquent, la transition sur l'événement s_2, à partir du sommet 3, ne peut pas être franchie puisque sa condition de garde "$x_1 = 0$" ne sera plus satisfaite, ce qui implique l'inconsistence du scénario de fonctionnement normal avec les observations générées par le système. Par contre, si nous estimons, à l'occurrence de s_2, l'ensemble de valeurs possibles pour x_1 dans le scénario de défaut, nous constatons que la valeur 0 est incluse dans l'intervalle des valeurs estimées. Ainsi, il est possible de franchir une transition sur l'événement observé s_2 à partir du sommet 6. En éliminant le scénario du fonctionnement normal, la procédure annonce l'occurrence du défaut fuite.

Lorsqu'un blocage de la vanne V_1 se produit, aucun autre événement ne sera observé. En se focalisant sur la durée du temps d'attente qui s'est écoulée sans avoir observé l'événement s_1, nous pouvons déterminer l'occurrence de ce défaut. Dans ce cas, à partir de l'estimation des valeurs possibles pour x_1, nous constatons que le système ne peut pas évoluer dans les sommets 1 ou 4 au delà d'une durée déterminée, sinon la condition d'invariant sera violée. En effet, après l'écoulement d'une certaine durée de temps, le seul état discret possible du système correspond au sommet final 7, qui est un état de défaillance.

5.4 Une démarche de diagnostic en-ligne pour les SDH

Nous présentons dans la suite les différents étapes de notre demarche de diagnostic pour les SDH, introduite dans (Derbel et al., 2009b,a). Dans un premier lieu, nous caractérisons les spécifications et les hypothèses retenues sur le modèle automate hybride rectangulaire du système à diagnostiquer. En second lieu, nous présentons la procédure de diagnostic des défauts, qui constitue la base de notre contribution.

5.4.1 Modèle hybride pour le diagnostic : spécifications et hypothèses

Dans notre travail, nous avons choisi le formalisme des automates hybrides rectangulaires comme un cadre de modélisation des SDH à diagnostiquer. L'application de notre approche de diagnostic nécessite la vérification de certaines spécifications et hypothèses sur le modèle du système. Les spécifications imposées permettent de caractériser la modélisation des défauts. Certaines de ces contraintes de modélisation ont été consi-

dérées dans le cadre de notre contribution de diagnostic à base d'automates temporisés, développée dans le chapitre précédent.

Le modèle automate rectangulaire hybride considéré dans notre approche de diagnostic, doit respecter les spécifications suivantes :

- le modèle est "complet". En effet, il décrit le comportement normal et les comportements anormaux (défaillants) du système ;
- nous désignons par Σ_f l'ensemble des événements de défauts. Nous supposons que tous les défauts considérés sont non observables, ainsi $\Sigma_f \subseteq \Sigma_{uo}$;
- nous supposons que l'ensemble des défauts Σ_f forme une partition de m sous-ensembles de défauts (ou modes de défaillance) $\Sigma_f = F_1 \cup \cdots \cup F_m$;
- les défauts sont permanents. En effet, le système ne peut pas retourner d'un mode de défaillance à un mode de fonctionnement normal ;
- nous ne considérons pas le scénario de défauts multiples : en effet, deux défauts de différents modes ne peuvent pas se produire successivement dans une même exécution du système ;
- nous supposons que les états initiaux correspondent à des états de fonctionnement normal du système ;
- nous supposons qu'il existe une partition de l'ensemble des sommets du système, où chaque sommet correspond à un mode de fonctionnement normal ou un mode de défaillance F_i, $i \in \{1, \ldots, m\}$.

Dans la dernière spécification, il s'agit d'établir une partition sur l'ensemble des sommets de l'automate hybride rectangulaire du système. En effet, nous définissons la fonction de répartition (sommet/mode de fonctionnement) $\varphi : Q \mapsto \{0, \ldots, m\}$ qui permet d'associer à chaque sommet q, un entier $i \in \{1, \ldots, m\}$, si le sommet q est accessible par une exécution contenant un défaut de l'ensemble F_i, sinon elle lui attribut la valeur 0. Cependant, un sommet peut être accessible par deux exécutions correspondant à deux différents modes de fonctionnement différents (voir l'exemple dans la figure 5.8). Afin d'éviter un tel cas d'ambiguïté, nous appliquons un ensemble de transformations sur le modèle du système afin qu'on puisse établir cette partition sur l'ensemble de ses sommets.

Dans la figure 5.8, les sommets 2, 3 et 4 sont accessibles à la fois par des exécutions normales ; i.e., ne contenant pas des événements de défauts, et des exécutions contenant le défaut f.

La transformation que nous désirons appliquer sur le modèle AHR du système consiste à dupliquer un ensemble de sommets et de transitions. Il s'agit d'éclater chaque partie commune entre deux chemins de l'automate qui correspondent à deux différents modes de fonctionnement. Dans la figure 5.9, nous illustrons l'AHR obtenu après l'introduction de ces transformations sur l'automate présenté dans la figure 5.8. Au cours de cette transformation, nous avons dupliqué la partie $2 \rightarrow 3 \rightarrow 4$, commune entre le chemin du

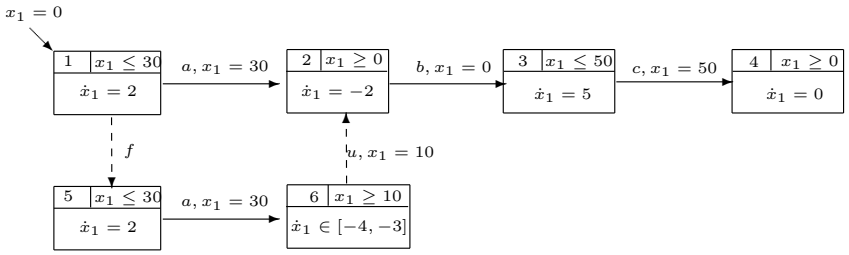

Fig. 5.8 – Sommets accessibles par des exécutions correspondant à de différents modes de fonctionnement

comportement normal et le chemin du comportement défaillant (contenant une transition sur l'événement f). Après ces transformations et en considérant que $\Sigma_f = F_1 = \{f\}$, la fonction de répartition φ est définie comme suit :

$$\varphi(q) = \begin{cases} 0, & \text{si } q \in \{1, 2, 3, 4\}\,; \\ 1, & \text{sinon.} \end{cases}$$

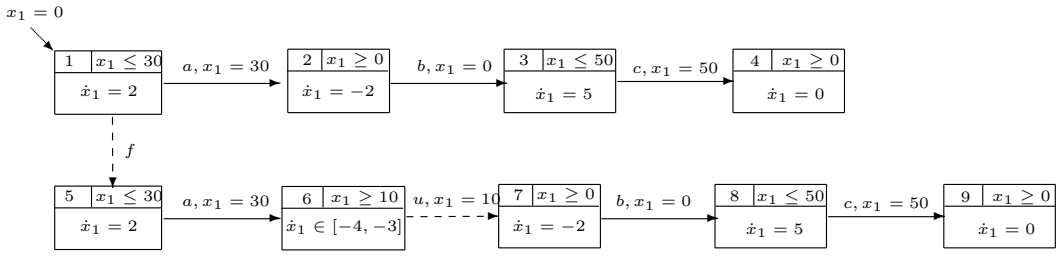

Fig. 5.9 – Modèle obtenu après la l'application de la transformation

Nous notons par H_0 la restriction d'un AHR H, qui décrit son comportement normal. Cette restriction peut être obtenue en éliminant du modèle initial H tout sommet q vérifiant la condition $\varphi(q) \neq 0$. Par ailleurs, l'élimination d'un sommet implique l'élimination de toute transition issue de, ou vers, ce sommet. Dans la figure 5.10, nous illustrons la restriction du comportement normal de l'AHR présenté dans la figure 5.9. L'AHR H_0 correspond à la partie de l'automate encadrée en pointillés.

Hypothèses : nous supposons que le modèle H vérifie les hypothèses suivantes :

$\mathcal{H}_1$: H est fortement non-zénon.

$\mathcal{H}_2$: la restriction du fonctionnement normal H_0 est un AHR initialisé.

La première hypothèse permet de garantir la divergence du temps dans les exécutions infinies. Ainsi, cette hypothèse garantit la progression du temps de chaque exécution du système. La seconde hypothèse stipule que la restriction du fonctionnement H_0 est un

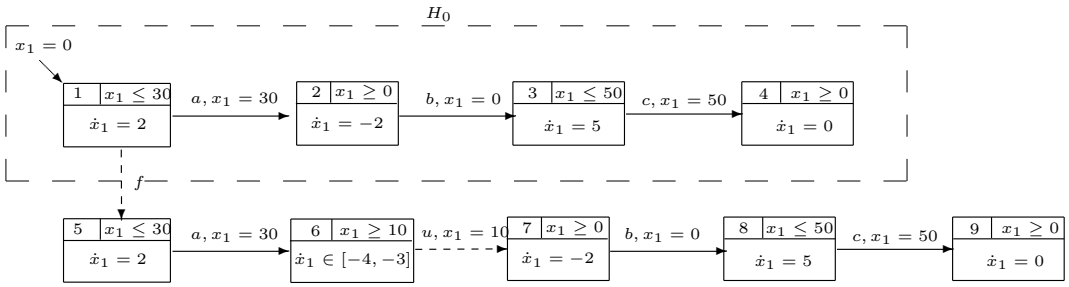

Fig. 5.10 – Restriction du comportement normal H_0

AHR initialisé. Nous rappelons que dans ce cas chaque changement de flux d'une variable entre deux sommets implique une réinitialisation de cette variable. La vérification de ces deux hypothèses est nécessaire pour l'application de notre méthode de test de la diagnosticabilité, tandis que l'application de notre procédure de diagnostic nécessite uniquement la vérification de la première hypothèse.

Nous allons vérifier ces hypothèses sur le modèle du système de chauffage de liquides décrit dans l'exemple 5.3.1. L'automate hybride rectangulaire présenté dans la figure 5.7 vérifie les spécifications énumérées ci-dessus, si l'on considère la partition de défauts $\Sigma_f = F_1 \cup F_2$, où $F_1 = \{fuite\}$ et $F_2 = \{blocage\_V_1\}$. La fonction de répartition φ est définie comme suit :

$$\varphi(q) = \begin{cases} 0, & \text{si } q \in \{1, 2, 3\}, \\ 1, & \text{si } q \in \{4, 5, 6\}, \\ 2, & \text{si } q \in \{7\}. \end{cases}$$

Dans ce qui suit, nous vérifions que le modèle considéré satisfait les hypothèses $\mathcal{H}_1$ et $\mathcal{H}_2$:

1. $\mathcal{H}_1$ est **satisfaite** par le modèle. Il suffit de remarquer que dans les deux cycles du modèle, l'horloge x_2 (1) admet la dynamique $\dot{x}_2 = 1$ dans tous les sommets, (2) est remise à zéro dans les transitions sur l'événement s_2 et (3) admet la valeur 40 comme une borne inférieure dans la garde de chaque transition associée à l'événement u (ainsi, dans chaque exécution d'un cycle s'écoule au moins 40 u.t.).
2. $\mathcal{H}_2$ est **satisfaite** par le modèle. En effet, le modèle du comportement normal, constitué par les sommets 1, 2 et 3 correspond à un AHR initialisé.

5.4.2 Procédure de diagnostic en-ligne

Tout au long de cette partie, nous supposons que le système à diagnostiquer est modélisé par un automate hybride rectangulaire $H = (L, X, \Sigma, E, inv, flux, init)$, vérifiant

les spécifications et les hypothèses énumérées ci-dessus.

Notre démarche de diagnostic pour les SDH repose sur l'exécution en-ligne d'une procédure de diagnostic qui permet d'estimer l'état courant du système à travers l'analyse du comportement discret observable du système. A partir des états estimés, une fonction de décision émet l'un des diagnostics suivants :

- "Défaillance F_i", $i \in \{1, \ldots, m\}$: un défaut de l'ensemble F_i s'est produit ;
- "Normal" : aucun défaut ne s'est produit ;
- "État uncertain" : aucune décision certaine ne peut être élaborée.

Nous soulignons que la procédure de diagnostic ne doit pas générer des fausses alarmes ; i.e., elle ne doit pas annoncer le diagnostic "Défaillance F_i", alors qu'aucun défaut de l'ensemble F_i ne s'est pas produit. De même, elle ne doit pas annoncer que le système est "Normal" alors qu'un défaut s'est produit.

Dans ce qui suit, nous illustrons intuitivement le schéma de fonctionnement de la procédure de diagnostic. Ensuite, nous présentons formellement cette procédure.

La procédure de diagnostic évolue comme une **machine d'état**, où l'état courant noté d_{cour}, fournit une **estimation de l'état courant** du système. Le calcul d'une nouvelle estimation est déclenché par :

1. l'observation d'un nouvel événement ;
2. le dépassement d'un délai seuil d'attente, sans aucun événement observé.

Dans le premier cas ; i.e., observation d'un événement $\sigma \in \Sigma_o$, la procédure de diagnostic estime l'état courant du système d_{cour} en fonction (1) du délai écoulé depuis la dernière estimation et (2) de l'occurrence de toute possible séquence d'événements non observables suivie par l'événement σ.

La procédure utilise un temporisateur, noté t pour mesurer le temps écoulé depuis la dernière estimation effectuée. Dans le second cas, ce temporisateur atteint une valeur seuil, sans que la procédure n'observe un nouvel événement. En effet, une fois qu'il atteint cette valeur seuil, appelée *Délai d'Attente Maximal* (DAM), le temporisateur expire et la procédure effectue une nouvelle estimation de l'état courant du système.

Il faut noter que la valeur DAM est déterminée dynamiquement au début de chaque cycle d'exécution de la procédure. Cette valeur est calculée en fonction de la dernière estimation de l'état du système, de la manière suivante : à partir de la dernière estimation de l'état du système d_{cour}, quel est le délai d'attente au plus tôt DAM, pour que les états estimés après l'écoulement de ce délai et/ou l'occurrence d'événements non observables soient F_i-certain, $i \in \{1, \ldots, m\}$; i.e., chaque état estimé est accessible par une exécution comportant un défaut de l'ensemble F_i. En effet, suite à l'écoulement de DAM u.t. depuis la dernière estimation, la procédure estime l'état du système à $t = DAM$ puis annonce

l'occurrence d'un défaut de l'ensemble F_i.

Cependant, il est possible qu'on ne réussit pas à retrouver un délai fini au bout duquel les états estimés sont F_i-certain. Par exemple, nous pouvons considérer le cas où le système opère indéfiniment dans le mode normal ; i.e., aucun défaut n'affecte son comportement nominal, et par conséquent, une valeur finie pour le délai DAM ne peut pas exister. Nous proposons alors d'explorer l'existence de ce délai dans une fenêtre temporelle bornée, appelée *fenêtre temporelle d'observation*. Cette fenêtre est donnée sous la forme d'un ensemble d'entiers $\{1, \ldots, T\}$, où l'entier $T \geq 1$ correspond à la borne supérieure de cette fenêtre. Cet entier doit être fournie comme un paramètre de la procédure.

Si les états estimés après l'écoulement de T u.t. ne sont pas F_i-certain, le délai DAM sera affecté à T. A l'expiration du temporisateur, la procédure de diagnostic effectuera dans ce cas une mise à jour de l'estimation d'état du système, sans annoncer l'occurrence d'un défaut.

Le diagramme illustré dans la figure 5.11, décrit d'une manière informelle, les différentes étapes réalisées pendant l'exécution de la procédure de diagnostic.

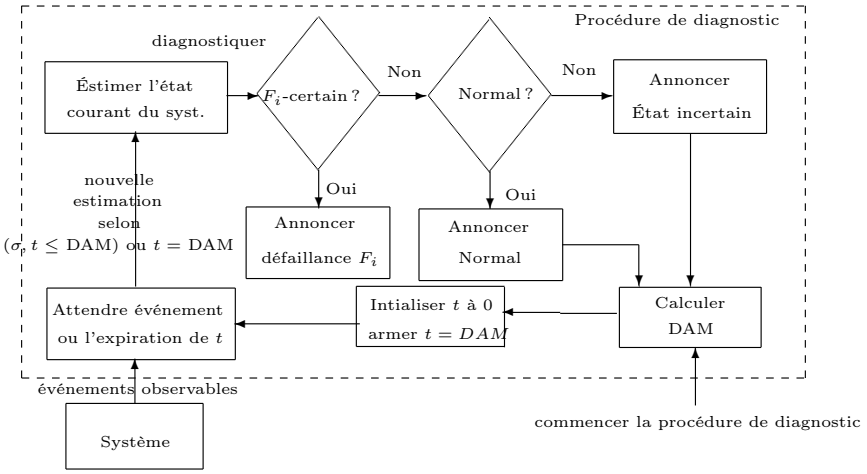

Fig. 5.11 – *Diagramme d'exécution de la procédure de diagnostic.*

Pas 1 : Calculer le délai d'attente maximal(DAM).
Pas 2 : Initialiser le temporisateur t à 0, et armer son expiration à $t = DAM$.
Pas 3 : Attendre l'occurrence d'un événement ou l'expiration du temporisateur t.
Pas 4 : Si un événement est observé à $t \leq DAM$ **ou** le temporisateur expire à $t = DAM$ **alors** estimer l'état courant du système d_{cour}.
Pas 5 : Evaluer l'estimation courante de l'état du système.

- **Si** les états estimés sont tous F_i-incertain **alors** annoncer "Défaillance F_i" ;
- **Sinon si** les états estimés sont normaux **alors** annoncer "Normal" ;
- **Sinon** annoncer "État incertain".

5.4.3 Elaboration formelle de la procédure de diagnostic en-ligne

Dans la suite, nous présentons une formalisation de notre procédure de diagnostic. Nous utilisons la représentation d'état symbolique pour l'implémentation de cette procédure. Nous introduisons d'abord quelques notions préliminaires. Ensuite, nous présentons l'algorithme de la procédure de diagnostic. Enfin, un exemple d'exécution de la procédure est présenté.

Nous notons par $\Phi = \{\mathcal{N}, \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m\}$, l'ensemble des étiquettes de diagnostic. Ces étiquettes permettent de suivre les défauts produits avant que le système n'évolue vers un état donné. Un état du système est dit *normal*, s'il est associé à l'étiquette $\mathcal{N}$ et dit état *F_i-certain*, $i \in \{1, \ldots, m\}$, s'il est associé à l'étiquette $\mathcal{F}_i$.

Nous considérons la même définition pour l'opérateur de propagation d'étiquettes de diagnostic $\otimes$, introduite dans le chapitre précédent.

Un *état de diagnostic* est défini par un ensemble de triplets, $\{[(q_k, z_k), \phi_k], k \in \{1, \ldots, k\}\}$, où $q_k \in Q$, $z_k \in \Psi(X)$ et $\phi_k \in \Phi$. Ainsi, un état de diagnostic correspond à un ensemble de régions enrichies par des étiquettes de diagnostic. Nous notons par D l'ensemble des états de diagnostic.

Nous définissons la fonction $\acute{E}val : D \rightarrow \Phi \cup \{\perp\}$, qui permet d'évaluer un état de diagnostic donné. En effet, la fonction $\acute{E}val$ renvoie l'étiquette $\mathcal{F}_i$, $i \in \{1, \ldots, m\}$ (respect. l'étiquette $\mathcal{N}$), si tous les éléments de d contiennent l'étiquette $\mathcal{F}_i$ (respect. l'étiquette $\mathcal{N}$) et renvoie $\perp$ dans le cas échéant. Formellement, étant donné un état de diagnostic $d = \{[(q_1, z_1), \phi_1], \ldots, [(q_k, z_k), \phi_k]\}$,

$$\acute{E}val(d) = \begin{cases} \phi \in \Phi, & \text{si } d \neq \emptyset \text{ et } \phi_1 = \cdots = \phi_k = \phi, \\ \perp, & \text{sinon.} \end{cases}$$

Un état de diagnostic d est dit F_i-certain, $i \in \{1, \ldots, m\}$, si $\acute{E}val(d) = \mathcal{F}_i$; i.e., tous ses éléments sont F_i-certain.

Nous présentons dans la suite, la fonction d'*accessibilité non observable à temps borné*, notée $\mathcal{TR}$. La fonction $\mathcal{TR}$ permet de déterminer l'ensemble des états accessibles depuis un ensemble d'états de départ $d_{départ}$, suite à l'occurrence d'événements non observables et l'écoulement d'une durée de temps, donnée par le paramètre $\beta \geq 1$, depuis l'occurrence du dernier événement observable (lorsque aucun événement observable ne s'est produit, nous considérons cette durée à partir des états initiaux).

L'application de cette fonction nécessite l'ajout d'une horloge y au modèle AHR du système, permettant de mesurer le temps écoulé à partir de la dernière occurrence d'un événement observable. Cette horloge est réinitialisée à 0 dans chaque état initial et lors du franchissement de chaque transition sur un événement observable. Dans chaque état

accessible en utilisant la fonction $\mathcal{TR}$, la valeur de l'horloge y est égale à la valeur β. Les étiquettes de diagnostic sont propagées durant ce calcul en utilisant l'opérateur $\otimes$.

Nous rappelons que l'hypothèse H_1 garantit la divergence du temps dans toute exécution du système, ce qui garantit l'arrêt de cette fonction. En effet, la valuation de l'horloge y progresse d'une manière uniforme jusqu'à atteindre la valeur β, au bout d'un nombre fini d'itérations.

Remarque 6. Nous pouvons remarquer une similarité entre la fonction $\mathcal{TR}$ et la fonction d'accessibilité non-observable $\mathcal{UR}$ définie dans le chapitre 4. Cependant, cette fonction se distingue par le fait qu'elle calcule les états accessibles à travers des transitions non observables à un instant donné, tandis que $\mathcal{UR}$ calcule les états accessibles à tout instant.

Algorithm 5 Fonction d'accessibilité non-observable à temps-borné $\mathcal{TR}$

Entrées : $d_{départ} \in D,\ \beta \in \mathbb{N}$

Sorties : états accessibles après β u.t..

1: initialisation : $A\_TRAITER \leftarrow \{[Post_c((q,z)),\phi] \mid [(q,z),\phi] \in d_{départ}\}$, $ÉTATS\_ACCESSIBLES \leftarrow A\_TRAITER$;
2: **répéter**
3: retirer un élément $[(q,z),\phi]$ de $A\_TRAITER$;
4: **pour tout** $q \xrightarrow{\sigma_{uo}} q' \in E$, où $\sigma_{uo} \in \Sigma_{uo}$ **faire**
5: calculer $[(q',z'),\phi']$, où $(q',z') = Post((q,z), q \xrightarrow{\sigma_{uo}} q')$ et $\phi' = \phi \otimes \sigma_{uo}$;
6: ajouter $[(q',z'),\phi']$ à $ÉTATS\_ACCESSIBLES$, si $z' \neq \emptyset$;
7: ajouter $[(q',z') \wedge (y < \beta),\phi']$ à $A\_TRAITER$, si $z' \wedge (y < \beta) \neq \emptyset$;
8: **fin pour**
9: **jusqu'à** $A\_TRAITER = \{\}$;
10: $ÉTATS\_ACCESSIBLES\_TEMPS\_BORNÉ \leftarrow \{[(q,\tilde{z}),\phi],$ où $[(q,z),\phi] \in ÉTATS\_ACCESSIBLES$, $\tilde{z} = z \wedge (y = \beta)$ et $\tilde{z} \neq \emptyset\}$;
11: **retourner** $ÉTATS\_ACCESSIBLES\_TEMPS\_BORNÉ$.

La procédure de diagnostic en-ligne est formellement décrite dans l'algorithme 6. Nous commençons par initialiser l'état courant du diagnostiqueur d_{cour} par la région $[(q_0,z_0),\mathcal{N}]$, ensuite , nous exécutons une boucle infinie qui correspond au traitement effectué par notre procédure lors de chaque estimation. En effet, chaque exécution de cette boucle correspond à l'élaboration d'une nouvelle estimation de l'état courant du système. La variable y_{cour} permet de mesurer le temps écoulé entre la date d'occurrence du dernier événement observable et la date de la dernière estimation effectuée. Cette variable est remise à zéro lorsqu'un événement est observé et elle est incrémentée par la valeur du temporisateur t après chaque exécution de la boucle. A un instant donné, la durée de temps qui s'est écoulée depuis l'occurrence du dernier événement observable correspond à la valeur $y_{cour} + t$.

Algorithm 6 Procédure de diagnostic

Entrées : T

Initialisation : $d_{cour} \leftarrow \{[(q_0, z_0), \mathcal{N}]\}$ et $y_{cour} \leftarrow 0$;
boucler
  si $\acute{E}val(\mathcal{TR}(d_{cour}, y_{cour} + T)) \in \{\mathcal{N}, \perp\}$ **alors**
    DAM $\leftarrow T$;
  sinon
    DAM $\leftarrow min\{k \in \{1, \ldots, T\} \mid \acute{E}val(\mathcal{TR}(d_{cour}, y_{cour} + k)) = \mathcal{F}_i, i \in \{1, \ldots, m\}\}$;
  finsi
  Réinitialiser le temporisateur t à 0 ;
  Armer l'expiration du temporisateur $t =$ DAM ;
  Attendre l'occurrence d'un événement $\sigma \in \Sigma_o$ ou l'expiration de t ($t =$ DAM) ;
  $y_{cour} \leftarrow y_{cour} + t$;
  Estimer l'état courant du système après l'écoulement de t u.t., $d_{cour} \leftarrow \mathcal{TR}(d_{cour}, y_{cour})$;
  si un événement $\sigma \in \Sigma_o$ est observé **alors**
    Estimer l'état courant suite à l'occurrence de σ, $d_{cour} \leftarrow \{[(q'_i, z'_i), \phi_i]$ tel que $(q'_i, z'_i) = Post_d((q_i, z_i), \xrightarrow{\sigma})$, pour chaque $(q_i, z_i, \phi_i) \in d_{cour}\}$;
    $y_{cour} \leftarrow 0$;
  finsi
  si $\acute{E}val(d_{cour}) = \mathcal{F}_i, i \in \{1, \ldots, m\}$ **alors**
    Annoncer l'occurrence d'une "Défaillance F_i" ;
    Arrêt de l'algorithme.
  sinon si $\acute{E}val(d_{cour}) = \mathcal{N}$ **alors**
    Annoncer "Normal", le système opère dans le mode normal ;
  sinon
    Annoncer "État incertain" ;
  finsi
fin boucle

Nous présentons dans la suite un exemple illustrant l'exécution de la procédure de diagnostic.

Exemple 5.4.1
Nous considérons deux cas d'application de la procédure de diagnostic sur le système de chauffage de liquides représenté dans la figure 5.3.1. Dans chaque cas, nous illustrons les étapes de l'exécution de la procédure de diagnostic étant donnée une séquence d'observations générée par le système. Nous rappelons que l'application de notre procédure nécessite l'introduction de l'horloge y dans le modèle du système, qui permet de déterminer le temps qui s'est écoulé depuis l'occurrence du dernier événement observable. Nous considérons la partition de défauts suivante : $\Sigma_f = F_1 \cup F_2$, où $F_1 = \{fuite\}$ et $F_2 = \{blocage\_V_1\}$.

Dans le premier cas d'application, nous supposons que le système effectue une exécution sur la trace "$\omega_1 = 125, s_1, 40, u, 70, s_2, 10, fuite, 115, s_1, 40, u, 55, s_2$". Nous fixons la valeur du paramètre T à 80. Nous présentons dans la suite les étapes de l'exécution de la procédure, lorsque le système génère la partie observable de ω_1 ; i.e., "$P(\omega_1) = 125, s_1, 110, s_2, 125, s_1, 95, s_2$".

1. Initialement, $d_{cour} := \{[(1, \langle x_1 \in [0, 500] \wedge x_2 = y = 0\rangle), \mathcal{N}]\}$;
2. Le délai DAM est estimé à 80 u.t.. Le temporisateur t expire après 80 u.t., $d_{cour} \leftarrow \{[(1, \langle x_1 \in [240, 500] \wedge x_2 = 80 \wedge y = 80\rangle), \mathcal{N}], [(4, \langle x_1 \in [160, 500] \wedge x_2 = 80 \wedge y = 80\rangle), \mathcal{F}_1], [(7, \langle x_1 \in [0, 500] \wedge x_2 = 80 \wedge y = 80\rangle), \mathcal{F}_2]\}$. La procédure génère le diagnostic "État incertain" ;
3. $DAM \leftarrow 80$. L'événement s_1 est observé à $t = 45$, $d_{cour} \leftarrow \{[(2, \langle x_1 = 500 \wedge x_2 = 0 \wedge y = 0\rangle), \mathcal{N}], [(5, \langle x_1 = 500 \wedge x_2 = 0 \wedge y = 0\rangle), \mathcal{F}_1]\}$. La procédure génère le diagnostic "État incertain" ;
4. $DAM \leftarrow 80$. Le temporisateur expire à $t = 80$, $d_{cour} \leftarrow \{[(3, \langle x_1 \in [180, 260] \wedge x_2 = 80 \wedge y = 80\rangle), \mathcal{N}], [(6, \langle x_1 \in [60, 140] \wedge x_2 = 80 \wedge y = 80\rangle), \mathcal{F}_1]\}$. La procédure génère le diagnostic "État incertain" ;
5. $DAM \leftarrow 80$. L'événement s_2 est observé à l'instant $t = 30$, $d_{cour} \leftarrow \{[(1, \langle x_1 = 0 \wedge x_2 = 110 \wedge y = 0\rangle), \mathcal{N}]\}$. La procédure génère le diagnostic "Normal" ;
6. $DAM \leftarrow 80$. Le temporisateur expire à $t = 80$, $d_{cour} := \{[(1, \langle x_1 \in [240, 400] \wedge x_2 = 190 \wedge y = 80\rangle), \mathcal{N}], [(4, \langle x_1 \in [260, 320] \wedge x_2 = 190 \wedge y = 80\rangle), \mathcal{F}_1], [(7, \langle x_1 \in [0, 400] \wedge x_2 = 190 \wedge y = 80\rangle), \mathcal{F}_2]\}$. La procédure génère le diagnostic "État incertain" ;
7. $DAM \leftarrow 80$. L'événement s_1 est observé à $t = 45$, $d_{cour} := \{[(2, \langle x_1 = 500 \wedge x_2 = 0 \wedge y = 0\rangle), \mathcal{N}], [(5, \langle x_1 = 500 \wedge x_2 = 0 \wedge y = 0\rangle), \mathcal{F}_1]\}$. La procédure génère le diagnostic "État incertain" ;

8. $DAM \leftarrow 80$. Le temporisateur expire à $t = 80$, . La procédure génère le diagnostic "État incertain" ;
9. $DAM \leftarrow 80$. L'événement s_2 **est observé à** $t = 15$, Les états estimés après l'écoulement de 15 u.t. est $d_{cour} := \{[(3, \langle x_1 \in [60, 150] \wedge x_2 = 95 \wedge y = 95\rangle), \mathcal{N}], [(6, \langle x_1 \in [0, 20] \wedge x_2 = 95 \wedge y = 95\rangle), \mathcal{F}_1]\}$. Il est clair que la transition discrète sur l'événement s_2 ne peut pas être franchie à partir de la région $(3, \langle x_1 \in [60, 150] \wedge x_2 = 95 \wedge y = 95\rangle)$ puisque sa condition de garde $x_1 = 0$ ne peut pas être satisfaite par aucune des valuations estimées. Ainsi, l'état courant estimé après le calcul du successeur discret sur l'événement s_2 est : $d_{cour} := \{[(4, \langle x_1 = 0 \wedge x_2 = 95 \wedge y = 0\rangle), \mathcal{F}_1]\}$. La procédure génère alors une alarme "Défaillance F_1", ensuite, l'algorithme s'arrête ;

Dans le deuxième cas d'application, nous supposons que le système effectue une exécution sur la trace "$\omega_2 = 127.2, s_1, 40, u, 70, s_2, 10, blocage\_V_1, 1, 1, 1, 1 \ldots$". Le paramètre T est maintenu à 80 u.t.. Nous présentons dans la suite quelques traces d'exécution de la procédure quand la séquence $P(\omega_2) =$ "$127.2, s_1, 110, s_2, 1, 1, 1, \ldots$" est générée par le système.

1. Initialement, $d_{cour} := \{[(1, \langle x_1 \in [0, 500] \wedge x_2 = y = 0\rangle), \mathcal{N}]\}$.
2. $DAM \leftarrow 80$. Le temporisateur t expire à $t = 80$, $d_{cour} \leftarrow \{[(1, \langle x_1 \in [240, 500] \wedge x_2 = 80 \wedge y = 80\rangle), \mathcal{N}], [(4, \langle x_1 \in [160, 500] \wedge x_2 = 80 \wedge y = 80\rangle), \mathcal{F}_1], [(7, \langle x_1 \in [0, 500] \wedge x_2 = 80 \wedge y = 80\rangle), \mathcal{F}_2]\}$. La procédure génère le diagnostic "État incertain" ;
3. $DAM \leftarrow 80$. L'événement s_1 est observé à $t = 47.2$, $d_{cour} \leftarrow \{[(2, \langle x_1 = 500 \wedge x_2 = 0 \wedge y = 0\rangle), \mathcal{N}], [(5, \langle x_1 = 500 \wedge x_2 = 0 \wedge y = 0\rangle), \mathcal{F}_1]\}$. La procédure génère le diagnostic "État incertain".
4. $DAM \leftarrow 80$. Le temporisateur expire à $t = 80$, $d_{cour} \leftarrow \{[(3, \langle x_1 \in [180, 260] \wedge x_2 = 80 \wedge y = 80\rangle), \mathcal{N}], [(6, \langle x_1 \in [60, 140] \wedge x_2 = 80 \wedge y = 80\rangle), \mathcal{F}_1]\}$. La procédure génère le diagnostic "État incertain".
5. $DAM \leftarrow 80$. L'événement s_2 est observé à l'instant $t = 30$, $d_{cour} \leftarrow \{[(1, \langle x_1 = 0 \wedge x_2 = 110 \wedge y = 0\rangle), \mathcal{N}]\}$. La procédure génère le diagnostic "Normal".
6. $DAM \leftarrow 80$. Le temporisateur expire à $t = 80$, $d_{cour} := \{[(1, \langle x_1 \in [240, 400] \wedge x_2 = 190 \wedge y = 80\rangle), \mathcal{N}], [(4, \langle x_1 \in [160, 320] \wedge x_2 = 190 \wedge y = 80\rangle), \mathcal{F}_1], [(7, \langle x_1 \in [0, 400] \wedge x_2 = 190 \wedge y = 80\rangle), \mathcal{F}_2]\}$. La procédure génère le diagnostic "État incertain".
7. $DAM \leftarrow 80$. Le temporisateur expire à $t = 80$, $d_{cour} := \{[(1, \langle x_1 \in [480, 500] \wedge x_2 = 270 \wedge y = 160\rangle), \mathcal{N}], [(4, \langle x_1 \in [320, 500] \wedge x_2 = 270 \wedge y =$

$160\rangle), \mathcal{F}_1], [(7, \langle x_1 \in [0, 500] \wedge x_2 = 270 \wedge y = 160\rangle), \mathcal{F}_2]\}$. La procédure génère le diagnostic "État incertain".

8. $DAM \leftarrow 80$. Le temporisateur expire à $t = 80$, $d_{cour} := \{[(4, \langle x_1 \in [480, 500] \wedge x_2 = 350 \wedge y = 240\rangle), \mathcal{F}_1], [(7, \langle x_1 \in [0, 500] \wedge x_2 = 350 \wedge y = 240\rangle), \mathcal{F}_2]\}$. La procédure génère le diagnostic "État incertain".

9. Nous commençons par calculer le délai DAM. En déterminant l'état du système estimé après l'écoulement de $T = 80$ u.t., nous constatons qu'il est F_2-certain. En effet, nous cherchons le plus petit délai dans la fenêtre temporelle $\{1, \ldots, 80\}$, pour lequel l'état estimé reste F_2-certain. Le **délai** DAM calculé est **égale à 11 u.t.**. En effet, après l'écoulement de ce délai, l'estimation associée au sommet 4 sera écartée puisque l'invariant "$x_1 \leq 500$" correspondant à ce sommet sera violé.

   Lorsque le **temporisateur** t **expire à** $t = 11$, $d_{cour} := \{[(7, \langle x_1 \in [0, 500] \wedge x_2 = 361 \wedge y = 251\rangle), \mathcal{F}_2]\}$. La procédure génère une alarme "Défaillance F_2", ensuite, l'algorithme s'arrête.

Remarque 7. Afin d'assurer le bon fonctionnement de la procédure de diagnostic, le temps nécessaire pour l'exécution des traitements présents dans la boucle principale ne doit pas excéder un certain seuil. Pratiquement, cela suppose que temps maximal nécessaire pour le calcul d'une estimation doit être inférieur au délai minimal séparant deux événements observables. Dans ce cadre, le choix du paramètre T (borne supérieure de la fenêtre temporelle d'observation) est d'une grande importance.

D'un coté, plus que ce paramètre est grand, plus le temps de traitement est plus important. D'abord, le temps nécessaire pour l'exécution de la fonction $\mathcal{TR}$ (ligne 3 dans l'algorithme 6) peut augmenter en fonction du paramètre T (surtout lorsque le modèle comporte des cycles de transitions non observables). En plus, le temps nécessaire pour l'estimation du délai DAM augmente logarithmiquement en fonction du paramètre T, si l'on considère une recherche dichotomique de ce délai dans l'intervalle $\{1, \ldots, T\}$.

D'un autre coté, plus le paramètre T est petit, plus le nombre d'exécutions de la boucle principale est important. Cela implique un effort de calcul plus important puisque le nombre d'estimations sur l'expiration du temporisateur va augmenter. Pour conclure, le choix du paramètre T repose sur un compromis entre un temps de traitement important pour une seule boucle lorsque T est trop grand et un nombre d'exécutions élevé de la boucle principale entre deux événements observables lorsque T est trop petit.

Nous proposons une solution pratique permettant d'optimiser le choix du paramètre T. D'abord, nous initialisons ce paramètre par la moyenne des délais entre les événements observables. Ensuite, nous varions ce paramètre jusqu'à avoir expérimentalement une valeur optimale du paramètre T.

Dans les deux exemples étudiés, la procédure de diagnostic réussit à identifier l'occurrence du défaut. Notre procédure de diagnostic demeure applicable, même s'il s'agit d'un modèle non diagnosticable du système. Cependant, la consistence du résultat fourni dans ce dernier cas n'est pas garantie puisque la procédure est incapable d'identifier l'occurrence de tout mode de défaillance au bout d'un délai fini. En effet, dans un modèle non diagnosticable, il peut exister deux traces ayant la même projection observable et qui correspondent à deux modes de fonctionnement différents du système. En s'appuyant sur un modèle non diagnosticable, la procédure de diagnostic peut générer indéfiniment le diagnostic "état incertain" alors qu'un défaut s'est produit. Dans ce cas, il est impossible de distinguer le comportement observable du mode de défaillance correspondant à ce défaut et les comportements observables associés aux autres modes de fonctionnement du système. Cette conclusion nous amène à étudier, dans la section suivante, la diagnosticabilité du modèle utilisé pour le diagnostic.

5.5 Diagnosticabilité des automates hybrides rectangulaires

Nous présentons dans cette section une notion de diagnosticabilité de langages temporisés, en considérant une partition de plusieurs modes de défaillance, que nous appelons la diagnosticabilité à *Horizon de Temps Limité* (HTL). Ensuite, nous présentons une approche systématique permettant la vérification de la diagnosticabilité à HTL d'un (language temporisé accepté par un) AHR soumis aux conditions considérées dans notre démarche de diagnostic (Derbel et al., 2009d).

5.5.1 Diagnosticabilité à Horizon de Temps Limité

Soit L un langage temporisé accepté par un AHR H, vérifiant les conditions discutées dans la sous section 5.4.1. Nous associons à chaque ensemble de défauts F_i, $i \in \{1, \ldots, m\}$, un sous langage de L, noté L_i, tel que chaque trace dans L_i contient au moins un événement de l'ensemble F_i. Nous désignons par $L_0 \subseteq L$, le sous-ensemble de traces de L, ne contenant pas de défauts.

La propriété suivante découle de l'absence du scénario de défauts multiples dans notre modèle ; i.e, une trace ne peut pas contenir deux défauts appartenant à deux modes différents.

Propriété 2. L forme une partition de $m + 1$ sous-langages :

- $L = L_0 \cup L_1 \cup \cdots \cup L_m$.
- $L_i \cap L_j = \emptyset$, $\forall i, j \in \{0, \ldots, m\}, i \neq j$.

Nous définissons la *fonction d'horizon temporel* $\theta : \{1, \ldots, m\} \rightarrow \mathbb{N}$ qui associe pour chaque ensemble de défauts $F_i, i \in \{1, \ldots, m\}$, un entier positif $\theta(i)$, noté aussi par θ_i. Nous désignons par $L_i^{\theta_i}$, $i \in \{1, \ldots, m\}$, le sous-langage de L_i, où dans chaque trace de $L_i^{\theta_i}$, au moins θ_i u.t. se sont écoulées après l'occurrence d'un défaut de l'ensemble F_i. En effet, étant donnée une trace temporisée $\omega \in L_i^{\theta_i}$ telle que $\omega = \delta_0, \sigma_1, \delta_1, \sigma_2, \ldots, \sigma_j, \delta_j \ldots, \sigma_n, \delta_n, \ldots$, nous avons $\sum_{k \geq j} \delta_k \geq \theta_i$, où σ_j correspond à la première occurrence d'un événement de F_i dans ω.

Dans la suite, nous définissons la diagnosticabilité à horizon de temps limité pour une partition de m modes de défaillance et une fonction d'horizon temporel θ donnée.

Définition 39. Un langage temporisé L est dit diagnosticable dans un *Horizon de Temps Limité* (HTL), étant donnée une fonction d'horizon temporel θ :

$\forall i \in \{1, \ldots, m\}, \forall \omega \in L_i^{\theta_i}, \forall \omega' \in L$:

$$P(\omega) = P(\omega') \Longrightarrow \omega' \in L_i.$$

D'après la définition ci-dessus, un langage L est diagnosticable dans un HTL défini par la fonction θ, si toute paire de traces ω et ω' dans L, où ω contient un défaut de F_i et ω' ne contient aucun défaut de F_i, ont des projections observables différentes, suite à l'écoulement de θ_i u.t. depuis l'occurrence du défaut de F_i.

En effet, la diagnosticabilité à HTL permet de garantir l'identification de chaque défaut de l'ensemble $F_i, i \in \{1, \ldots, m\}$, au bout d'un délai θ_i u.t., à partir de son occurrence. Cela implique la discrimination du mode de défaillance F_i dans un intervalle de temps $[0, \theta_i]$ qui suit l'occurrence du défaut, appelé *fenêtre temporelle à horizon limité* (Ben Hadj-Alouane et al., 1994). L'identification du défaut se traduit par l'occurrence d'une séquence d'événements observables dans cette fenêtre de temps, permettant de distinguer d'une manière unique l'occurrence de ce défaut. La limitation de l'horizon du temps d'observation à un intervalle $[0, \theta_i]$, pour la vérification de la diagnosticabilité, justifie l'usage du terme diagnosticabilité à HTL.

Remarque 8. Nous soulignons que dans la définition de diagnosticabilité à HTL nous ne cherchons pas à déterminer l'existence d'une fonction d'horizon temporel pour lequel le langage est diagnosticable contrairement à la définition de diagnosticabilité donnée dans le chapitre 4. En effet, dans la définition 39, nous cherchons l'existence d'un entier positif Δ pour lequel le langage temporisé considéré est diagnosticable. Par contre, dans la définition ci-dessus, la fonction θ est préalablement fixée.

Exemple 5.5.1
Considérons le langage temporisé L accepté par l'AHR illustré dans la figure 5.12. Nous supposons que $\Sigma_o = \{a, b\}$ et $\Sigma_f = \{f_1, f_2\}$. Trois cas sont considérés dans cet exemple.

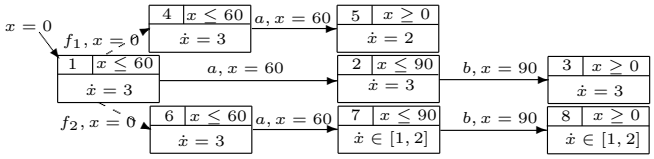

Fig. 5.12 – *Etude de la diagnosticabilité à HTL d'un AHR.*

Dans le **premier cas**, nous considérons la partition de défauts : $\Sigma_f = F_1$, avec $F_1 = \{f_1, f_2\}$ et $\theta_1 = 35$. Dans le fonctionnement normal du système, l'événement b doit être observé après le passage de $\frac{90}{3} = 30$ u.t. depuis l'état initial. Par contre, si un défaut de l'ensemble F_1 se produit, cet événement sera observé plus tard. Par conséquent, nous pouvons distinguer entre les comportements observables des deux modes de fonctionnement du système, dans l'intervalle $[0, 35]$ qui suit l'occurrence d'un défaut (f ne peut se produire qu'à l'instant $x = 0$). Nous concluons que L est diagnosticable dans un HTL.

Dans le **second cas**, nous considérons la partition de défauts suivantes : $\Sigma_f = F_1 \cup F_2$, où $F_1 = \{f_1\}$ et $F_2 = \{f_2\}$, et la fonction d'horizon temporel définie par $\theta_1 = 35$ et $\theta_2 = 35$. En considérant l'exemple des traces "$\omega = f_1, 20, a, 20$" et "$\omega' = f_2, 20, a, 20$", nous pouvons établir que L n'est pas diagnosticable dans l'HTL défini par θ, puisque ces deux traces admettent des projections observables identiques.

Dans le **troisième cas**, nous considérons la même partition de défauts du second cas, par contre, nous élargissons la fenêtre temporelle θ de la manière suivante : $\theta_1 = 60$ et $\theta_2 = 60$. L'événement b ne peut pas être observé dans aucune trace contenant un défaut de F_1. Pour toute trace contenant un défaut de F_2, l'événement b est observé plus tard que dans le cas de fonctionnement normal (il se produit dans l'intervalle de temps [35,50] après l'occurrence de f_2). Par conséquent, L est diagnosticable dans l'HTL défini par la fenêtre temporelle $\theta_1 = 60$ et $\theta_2 = 60$.

5.5.2 Vérification de la diagnosticabilité

Dans la suite, nous présentons une approche systématique permettant la vérification de la diagnosticabilité à HTL, pour les langages temporisés acceptés par des AHRs soumis aux conditions considérées dans notre démarche de diagnostic. Notre approche reprend quelques techniques inspirées des travaux introduits dans (Sampath et al., 1995; Tripakis, 2002). D'abord, nous détaillons le principe de notre procédure de vérification. Ensuite, nous présentons un théorème liant la diagnosticabilité à HTL avec l'application de cette procédure.

5.5.2.1 Procédure de vérification

Le principe de notre procédure de vérification de la diagnosticabilité à HTL repose sur l'exploration de toutes les paires de traces temporisées ω et ω', où $\omega \in L_i^{\theta_i}$, $\omega' \in L - L_i$ et $P(\omega) = P(\omega')$. L'existence d'une telle paire de traces implique la non diagnosticabilité de L, dans l'HTL défini par la fonction θ. Puisqu'il existe une infinité de paires de traces à vérifier, nous adoptons une démarche symbolique pour la résolution de ce problème. La procédure suivante énumère brièvement les différentes étapes de cette démarche.

Algorithm 7 Procédure *diagnosticable*

Entrées : AHR H, fenêtre temporelle θ
Sorties : {"Oui","Non"}

pour tout $i \in \{1, \ldots, m\}$ **faire**
  Construire les copies modifiées H_i et $H_{\bar{i}}$ de H ;
  Construire le produit $H_{i,\bar{i}} = H_i \otimes H_{\bar{i}}$ synchronisé sur les événements observables ;
  Construire le produit $H^{\theta}_{i,\bar{i}} = H_{i,\bar{i}} \otimes TB^{\theta}$;
  si $Accessible(H^{\theta}_{i,\bar{i}})$ ="Oui" **alors**
    retourner "Non" ;
  finsi
fin pour
retourner "Oui" ;

La **première étape** de la procédure consiste à construire pour chaque mode de défaillance F_i, $i \in \{1, \ldots, m\}$, une paire de AHRs, notés H_i et $H_{\bar{i}}$. L'AHR H_i accepte l'ensemble de traces de L ne contenant pas de défauts de l'ensemble F_j, pour $j \in \{1, \ldots, m\}$ et $j \neq i$. L'AHR $H_{\bar{i}}$ accepte toutes les traces de L à l'exception des traces contenant des défauts de l'ensemble F_i. Les AHRs H_i et $H_{\bar{i}}$ sont obtenus en appliquant les modifications suivantes sur deux copies de H.

L'AHR $H_i = (Q_i, X_i, \Sigma_i, E_i, inv_i, flux_i, init_i, M_i)$, $i \in \{1, \ldots, m\}$, est une copie de H sur laquelle on applique les modifications suivantes :

1. renommer chaque variable x_k par x_k^i ;
2. renommer chaque sommet q_k par q_k^i ;
3. éliminer tous les événements appartenant à $\Sigma_f - F_i$ de l'ensemble Σ_i, ainsi que toutes les transitions correspondantes à ces événements ;
4. renommer dans Σ_i, chaque événement σ_k de $\Sigma_{uo} - F_i$ par σ_k^i ;
5. mettre à jour les noms des variables, des événements et des sommets dans toutes les contraintes de gardes, d'invariants et de flux, en concordance avec les modifications ci-dessus ;
6. ajouter un chronomètre y^i tel que (1) y^i est réinitialisé à 0 dans les états initiaux,

(2) pour chaque sommet $q \in Q_i$, si $\varphi(q) = i$ alors nous ajoutons la condition $\dot{y}^i = 1$ à $flux_i(q)$, sinon, nous ajoutons la condition $\dot{y}^i = 0$.

L'AHR $H_{\bar{i}} = (Q_{\bar{i}}, X_{\bar{i}}, \Sigma_{\bar{i}}, E_{\bar{i}}, inv_{\bar{i}}, flux_{\bar{i}}, init_{\bar{i}}, M_{\bar{i}})$, $i \in \{1, \ldots, m\}$ est une copie de H sur laquelle nous introduisons les modifications suivantes :

1. éliminer de $\Sigma_{\bar{i}}$ tous les événements appartenant à F_i ainsi que toutes les correspondantes transitions ;
2. pour chaque $j \in \{1, \ldots, m\}$, $j \neq i$, nous ajoutons un chronomètre y^j tel que (1) y^j est réinitialisé 0 dans les états initiaux ; (2) pour chaque sommet $q \in Q_{\bar{i}}$, si $\varphi(q) = j$ alors nous ajoutons la condition $\dot{y}^j = 1$ à $flux_{\bar{i}}(q)$, sinon, nous ajoutons la condition $\dot{y}^j = 0$.

Nous remarquons qu'un chronomètre y^k, $k \in \{1, \ldots, m\}$ n'est activé dans un sommet q ; i.e., $[\![flux(q)]\!](\dot{y}^k) = 1$, que si le sommet q est associé au mode de défaillance F_k ($\varphi(q) = k$). Ensuite, ce chronomètre reste activé pour tous les successeurs de ce sommet, puisque les défauts considérées dans notre travail sont permanents. Ainsi, chaque chronomètre y^k mesure le temps qui s'est écoulé depuis la première occurrence d'un défaut de l'ensemble F_k.

Proposition 2. Nous notons par $L(H_i)$ et $L(H_{\bar{i}})$, les langages temporisés acceptés par les AHRs H_i et $H_{\bar{i}}$, respectivement. Pour $i \in \{1, \ldots, m\}$, les propositions suivantes découlent de la construction des AHRs H_i et $H_{\bar{i}}$:

- $L(H_i) = L - \bigcup_{k=1, k\neq i}^{m} L_k = L_0 \cup L_i$;
- $L(H_{\bar{i}}) = L - L_i$.

Dans la **seconde étape** de la procédure, nous construisons pour chaque mode de défaillance $i \in \{1, \ldots, m\}$, un produit synchrone de AHRs $H_{i,\bar{i}} = H_i \otimes H_{\bar{i}}$. Nous notons que $H_{i,\bar{i}}$ synchronise H_i et $H_{\bar{i}}$ seulement sur les événements observables, puisque $\Sigma_i \cap \Sigma_{\bar{i}} = \Sigma_o$. En effet, chaque trace acceptée par $H_{i,\bar{i}}$ correspond à deux traces ayant la même projection observable, l'une dans H_i et l'autre dans $H_{\bar{i}}$, comme établit la proposition suivant :

Proposition 3. Pour tout $i \in \{1, \ldots, m\}$, les propositions suivantes sont équivalentes :

1. Il existe une paire d'exécutions de H_i et $H_{\bar{i}}$, sur les traces temporisées ω_1 et ω_2, atteignant les états $(q_1, \mathbf{v_1})$ et $(q_2, \mathbf{v_2})$, tel que $P(\omega_1) = P(\omega_2)$;
2. Il existe une exécution de $H_{i,\bar{i}}$, sur une trace temporisée ω, atteignant l'état $((q_1, q_2), \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} \end{bmatrix})$.

L'existence d'une trace temporisée acceptée par $H_{i,\bar{i}}$ et contenant un défaut de l'ensemble F_k, implique l'existence d'une paire de traces de L_k et $L - L_k$, respectivement, ayant la même projection observable, comme établit le lemme suivant :

Lemme 7. (1) et (2) sont équivalents :

1. Il existe une exécution de $H_{i,\bar{i}}$, $i \in \{1, \ldots, m\}$, sur une trace temporisée ω, telle que ω contient un défaut de l'ensemble F_k.
2. Il existe une paire de traces temporisées $\omega_1 \in L_k$ et $\omega_2 \in L - L_k$, $i \in \{1, \ldots, m\}$, telle que $P(\omega_1) = P(\omega_2)$.

Preuve : voir annexe B.

Proposition 4. Nous notons par $H_i^0, H_{\bar{i}}^0$ et $H_{i,\bar{i}}^0$, les restrictions des comportements normaux de, respectivement, H_i $H_{\bar{i}}$ et $H_{i,\bar{i}}$, $i \in \{1, \ldots, m\}$. Ces AHRs sont obtenus par l'élimination de toutes les transitions sur des événements de défauts de ces AHRs, alors :

- $H_i^0 \otimes H_{\bar{i}}^0 = H_{i,\bar{i}}^0$ (preuve immédiate) ;
- $H_{i,\bar{i}}^0$ est un AHR initialisé (découle de l'hypothèse H_2 et la proposition 1).

Dans la **troisième étape** de la procédure, nous construisons l'AHR $H_{i,\bar{i}}^{\theta} = (Q_{i,\bar{i}}, X_{i,\bar{i}}, \Sigma_{i,\bar{i}}, E_{i,\bar{i}}, inv_{i,\bar{i}}, flux_{i,\bar{i}}, init_{i,\bar{i}}, M_{i,\bar{i}})$, qui correspond au produit de $H_{i,\bar{i}}$ avec un simple AHR à deux sommets, noté $TB^{\theta} = (\{q_{\overline{M}}, q_M\}, \{y^1, \ldots, y^m\}, \{\mu\}, E, inv, flux, init, \{q_M\})$ et illustré dans la figure 5.13. Nous remarquons l'absence de synchronisation d'événements au cours de ce produit puisque $\mu \notin \Sigma$. L'application de cette étape nous permet d'isoler l'ensemble des traces appartenant à $L_k^{\theta_k}$, pour $k \in \{1, \ldots, m\}$. En effet, la progression du temps est bloquée dans les exécutions de l'AHR $H_{i,\bar{i}}^{\theta}$ contenant des défauts de l'ensemble F_k, $k \in \{1, \ldots, m\}$ après l'expiration de fenêtre temporelle définie par l'intervalle $[0, \theta_i]$; i.e., lorsque $y^k = \theta_k$. Ainsi, lorsque le chronomètre y^k atteint la valeur θ_k dans une exécution de $H_{i,\bar{i}}^{\theta}$ contenant un défaut de F_k, $k \in \{1, \ldots, m\}$, une transition discrète vers un sommet marqué sera franchie. Ensuite, la condition d'invariant de ce sommet marqué sera violée instantanément après l'entrée dans ce sommet, impliquant le blocage de cette exécution.

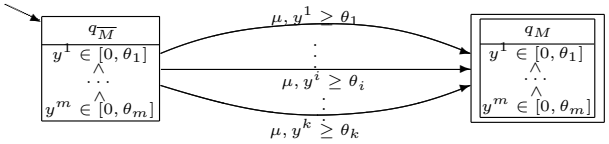

FIG. 5.13 – *L'AHR TB^{θ}*

La **dernière étape** de la procédure de vérification consiste à tester l'existence d'exécutions de $H_{i,\bar{i}}^{\theta}$, atteignant un sommet marqué, pour chaque $i \in \{1, \ldots, m\}$. Pour ce faire, nous utilisons la fonction d'analyse d'accessibilité illustrée dans l'algorithme 4.

Cette fonction utilise une analyse symbolique pour vérifier l'existence d'états marqués, accessible depuis l'ensemble des états initiaux, représentés par (q_0, z_0).

Le théorème suivant établit la convergence de la procédure d'analyse d'accessibilité, appliquée sur l'AHR

Théorème 2. La fonction d'analyse d'accessibilité, appliquée aux AHRs $H^{\theta}_{i,\bar{i}}$, pour tout $i \in \{1, \ldots, m\}$, s'arrête au bout d'un temps fini.

Intuitivement, ce résultat peut être expliqué dans les deux points suivants. D'une part, la restriction du comportement normal de l'AHR $H^{\theta}_{i,\bar{i}}$ est un AHR initialisé qui est une sous-classe décidable pour le problème d'analyse d'accessibilité (Henzinger et al., 1998). En effet, la procédure d'analyse s'arrête au bout d'un temps fini en parcourant cette restriction. D'autre part, l'analyse de la partie restante de l'AHR (les comportements défaillants) s'arrête sous l'influence des conditions d'invariants définies sur les chronomètres y^i, $i \in \{1, \ldots, m\}$, dans les sommets associés aux comportements défaillants. Une preuve complète de ce théorème est donnée dans l'annexe B.

L'AHR $H^{\theta}_{i,\bar{i}}$, $i \in \{1, \ldots, m\}$ permet d'isoler toute paire de traces dans L_k et $L - L_k$, $k \in \{1, \ldots, m\}$, telle que θ_k u.t. se sont écoulées après la première occurrence d'un défaut de l'ensemble F_k. En effet, l'existence d'une trace temporisée dont l'exécution atteint un sommet marqué, dans un AHR $H^{\theta}_{i,\bar{i}}$, $i \in \{1, \ldots, m\}$, implique l'existence d'une paire de traces dans $L_k^{\theta_k}$ et $L - L_k$ ayant la même projection observable. Cette propriété est formulée dans le lemme suivant :

Lemme 8. Les propositions suivantes sont équivalentes :

1. Il existe un sommet marqué dans l'AHR $H^{\theta}_{i,\bar{i}}$, $i \in \{1, \ldots, m\}$, accessible depuis un état initial.
2. Il existe une paire de traces temporisées $\omega_1 \in L_k^{\theta_k}$ et $\omega_2 \in L - L_k$, $k \in \{1, \ldots, m\}$, telle que $P(\omega_1) = P(\omega_2)$.

Preuve : voir annexe B.

Le théorème suivant établit une relation entre l'accessibilité des sommets marqués dans l'AHR $H^{\theta}_{i,\bar{i}}$, et la diagnosticabilité à HTL de L.

Théorème 3. Le langage temporisé L est diagnosticable dans un HTL θ, si et seulement si, pour tout $i \in \{1, \ldots, m\}$, aucun sommet marqué n'est accessible dans l'AHR $H^{\theta}_{i,\bar{i}}$; i.e., $\forall i \in \{1, \ldots, m\} : \mathit{Accessible}(H^{\theta}_{i,\bar{i}}) =$"NON".

Preuve : le sens ($\Longrightarrow$) : nous supposons qu'un sommet marqué est accessible dans $H^{\theta}_{i,\bar{i}}$. En effet, $\exists q \in M_{i,\bar{i}}, i \in \{1, \ldots, m\}$, tel que le sommet marqué q est accessible à travers une exécution de $H^{\theta}_{i,\bar{i}}$ sur une trace temporisée ω. D'après le lemme 8, il existe

une paire de traces $\omega_1 \in L_k^{\theta_k}$ et $\omega_2 \in L - L_k$, $k \in \{1, \ldots, m\}$, telle que $P(\omega_1) = P(\omega_2)$. En conclusion, L n'est pas diagnosticable dans l'HTL θ.

Dans le sens opposé ($\Longleftarrow$), nous supposons que L n'est pas diagnosticable. Alors, il existe $k \in \{1, \ldots, m\}$ et une paire de traces $\omega_1 \in L_k^{\theta_k}$ et $\omega_2 \notin L_k$, telle que $P(\omega_1) = P(\omega_2)$. Puisque $\omega_2 \in L - L_k$ et d'après le lemme 8, nous pouvons conclure qu'il existe $i \in \{1, \ldots, m\}$ et une exécution de l'AHR $H^{\theta}_{i,\bar{i}}$, atteignant un sommet marqué.

□

Dans la suite, nous présentons un exemple d'application permettant la vérification de la diagnosticabilité du modèle considéré dans l'exemple 5.3.1.

Exemple 5.5.2

Nous présentons dans les figures 5.14, 5.15, 5.16, et 5.17 les différents AHRs nécessaires pour la vérification de la diagnosticabilité du modèle décrit dans la figure 5.7. Pour des raisons de clarté, nous n'avons pas présenté les AHRs $H^{\theta}_{1,\bar{1}}$ et $H^{\theta}_{2,\bar{2}}$. Ces AHRs peuvent être déduits facilement à partir des AHRs $H_{1,\bar{1}}$ et $H_{2,\bar{2}}$.

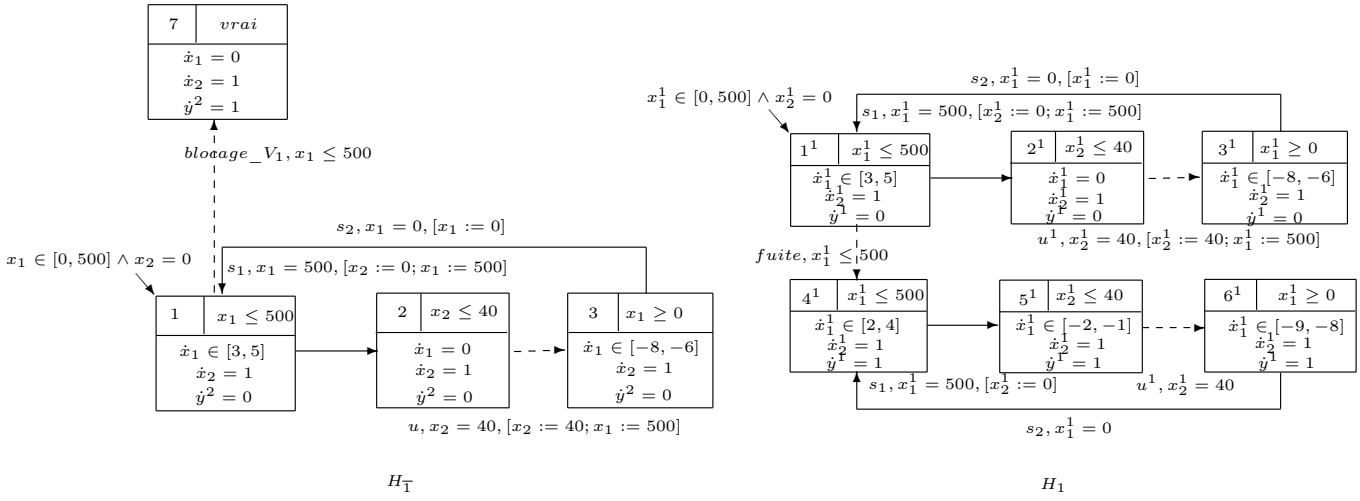

FIG. 5.14 – *Les AHRs H_1 et $H_{\bar{1}}$*

Nous considérons deux applications de la procédure de vérification de diagnosticabilité à HTL pour les fenêtres temporelles $\theta_1 = \theta_2 = 125$ et $\theta_1 = \theta_2 = 300$.

Dans le premier cas d'application, nous considérons $\theta_1 = \theta_2 = 125$. La procédure d'analyse d'accessibilité renvoie "OUI" lorsqu'elle est appliquée sur

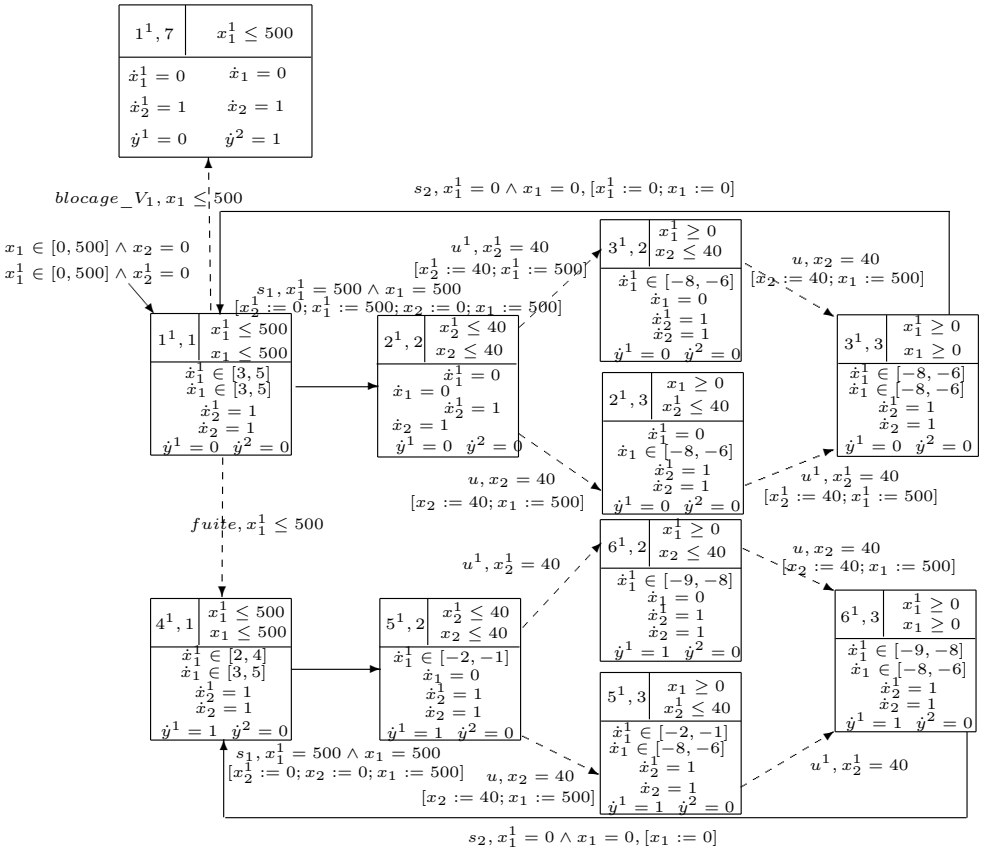

FIG. 5.15 – *L'AHR* $H_{1,\overline{1}}$.

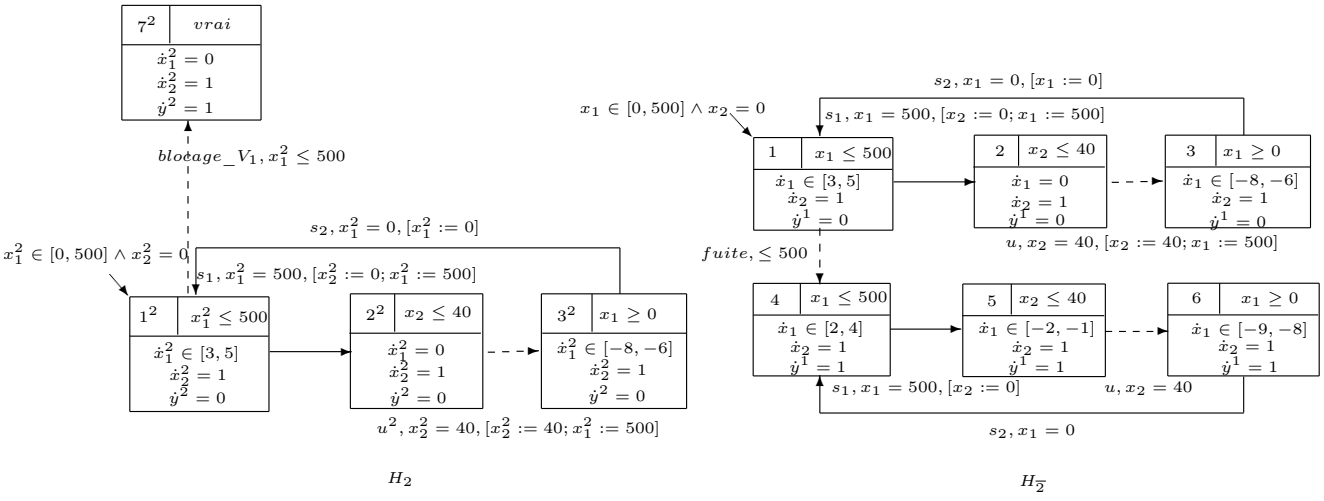

FIG. 5.16 – *Les AHRs* H_2 *et* $H_{\overline{2}}$

l'AHR $H^{\theta}_{1,\overline{1}}$. Par conséquent, il existe un sommet marqué accessible depuis un état initial et ainsi, le langage L n'est pas diagnosticable dans cette fenêtre temporelle. A titre d'exemple, nous pouvons retenir l'accessibilité de l'état

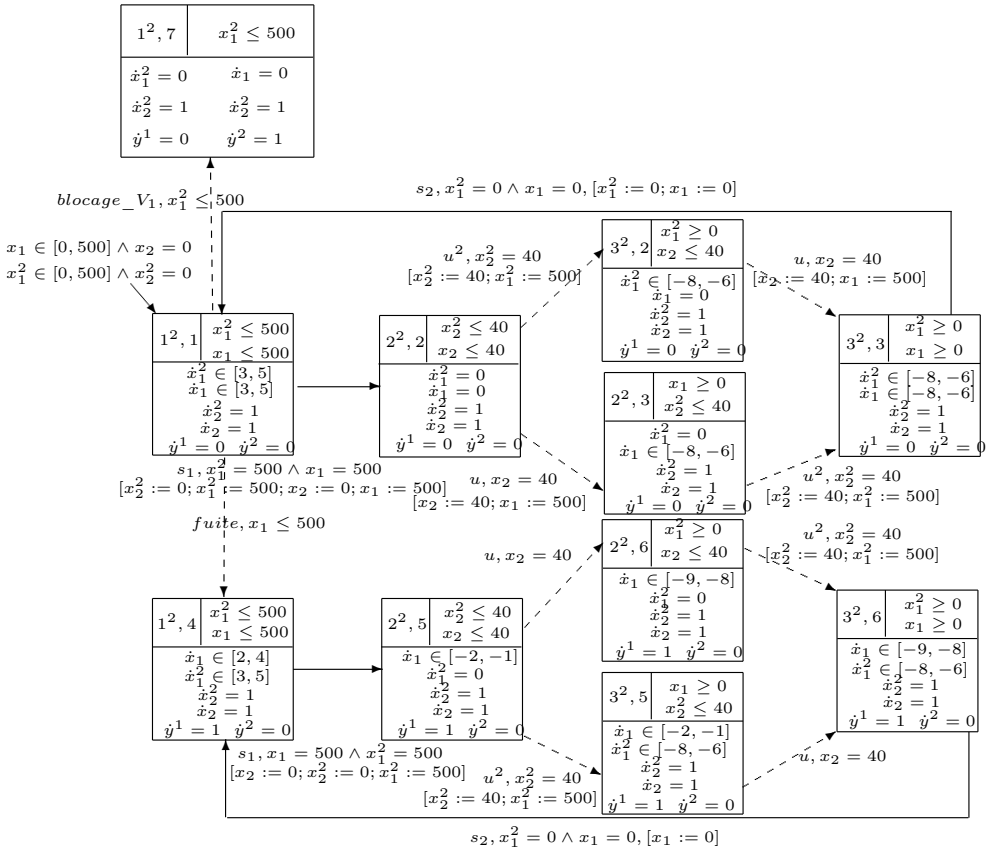

FIG. 5.17 – *L'AHR* $H_{2,\overline{2}}$.

marqué $((5^1, 2, q_M), \langle x_1 = x_1^1 = 500 \wedge x_2 = x_2^1 = y^2 = 0 \wedge y^1 = 125 \rangle)$ comme il est indiqué dans la figure 5.18. En effet, l'accessibilité de cet état implique l'existence de deux traces temporisées ayant la même projection observable, l'une contient le défaut $fuite$ (acceptée par H_1) et l'autre ne contenant aucun défaut (accepté par $H_{\overline{1}}$).

Nous considérons dans un deuxième cas d'application que $\theta_1 = \theta_2 = 300$, alors l'algorithme 4 renvoie "NON", indiquant qu'aucun sommet marqué n'est accessible depuis un état initial dans les AHRs $H^{\theta}_{1,\overline{1}}$ et $H^{\theta}_{2,\overline{2}}$. Par conséquence, L est diagnosticable dans le HTL défini par la fenêtre temporelle $\theta_1 = \theta_2 = 300$. Ainsi, chaque défaut pourra être identifié au bout de 300 u.t. de son occurrence.

5.6 Complexité

Dans un premier volet, nous présentons quelques remarques sur les issues d'implémentation de notre procédure de diagnostic et de notre démarche de vérification de la

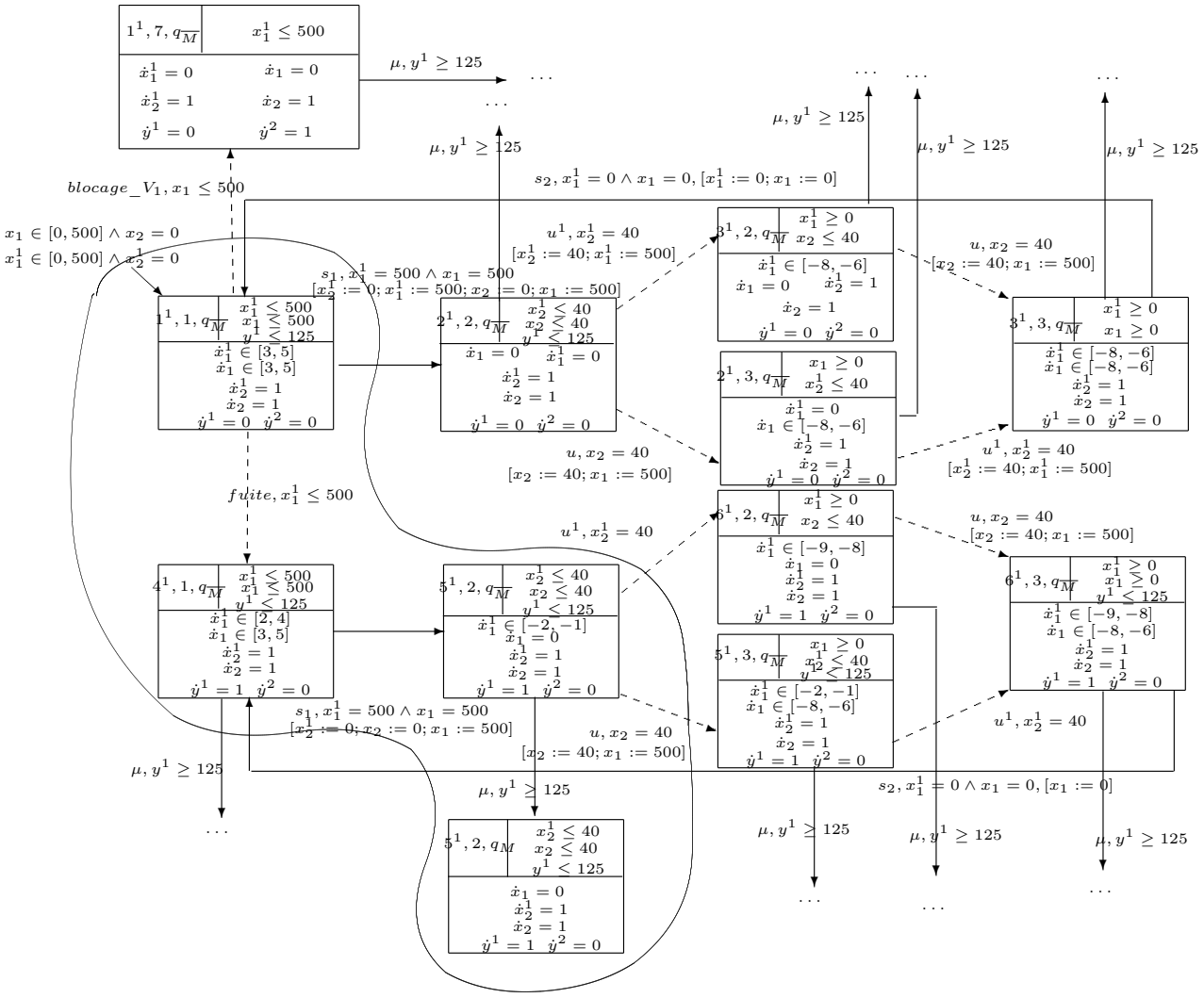

FIG. 5.18 – *Exemple de l'accessibilité d'un sommet marqué dans l'AHR* $H^{\theta}_{1,\overline{1}}$.

diagnosticabilité à HTL. Dans un second volet, nous présentons quelques éléments sur la complexité des algorithmes utilisés dans ce travail.

5.6.1 Quelques éléments sur l'implémentation de notre démarche de diagnostic

L'algorithme de la procédure de diagnostic ainsi que celui de la vérification d'accessibilité des AHRs reposent essentiellement sur des opérations de calcul de successeurs continus et de successeurs discrets de régions. Une région de l'espace d'état est représentée par une paire sommet/polyhèdre.

Les polyèdres peuvent être classés en deux ensembles : les polyèdres *convexes* et les polyèdres *non-convexes*. La première classe de polyèdres est la plus simple à utiliser. Les différentes opérations appliquées dans notre travail pour le calcul des successeurs d'une région : intersection, futur et réinitialisation ainsi que l'utilisation de contraintes de gardes et d'invariants rectangulaires, préservent la convexité des régions calculées. La manipulation de polyhèdres convexes dans les algorithmes développés dans notre travail facilite la tâche de calcul et réduit la taille des structures de données utilisées pour le stockage de ces régions.

Plusieurs outils de vérification d'automates hybrides linéaires développés ont été proposés dans la littérature, tels que PHAVer (Frehse, 2005) et HyTech (Henzinger et al., 1997). Ces outils implémentent des structures de données ainsi qu'un ensemble d'opérations élémentaires permettant d'analyser l'espace d'état des automates hybrides linéaires.

Dans le cadre de notre travail, nous pouvons utiliser la bibliothèque de manipulation de polyhèdres Parma (Bagnara et al., 2002) pour l'implémentation de la procédure de diagnostic. Cette bibliothèque, utilisée par l'outil PHAVer, présente une implémentation des régions polyhèdrales ainsi que les opérations employées dans notre travail pour leur manipulation. Cette implémentation est robuste puisqu'elle permet de supporter des valeurs arbitrairement larges. Afin d'appliquer notre procédure de vérification de diagnosticabilité, nous pouvons utiliser l'implémentation de la procédure d'analyse d'accessibilité développée dans l'outil PHAVer. Afin de forcer la convergence de l'algorithme d'analyse en avant, PHAVer applique des techniques d'approximation sur les régions polyhédrales calculées. Ces techniques d'approximation peuvent être désactivées lors de l'analyse des AHRs considérés dans notre travail puisque la convergence de la procédure d'analyse d'accessibilité est garantie pour ces AHRs.

5.6.2 Étude de la complexité

La procédure de diagnostic est un semi-algorithme qui exécute indéfiniment une boucle permettant d'estimer d'état courant du système. L'exécution de cette boucle est déclenchée par l'observation d'un événement ou le dépassement d'un délai seuil d'attente.

La complexité d'exécution de cette procédure de diagnostic est théoriquement exponentielle en fonction du nombre d'observations reçues et de la taille du modèle AHR du système. Par ailleurs, l'exécution de la procédure de diagnostic nécessite théoriquement un espace infini de mémoire. Ceci est attendue puisqu'il s'agit d'un algorithme en-ligne qui effectue un calcul incrémental. Cependant, cette complexité théorique ne reflète pas les besoins réels des algorithmes utilisés. En plus, il est possible d'employer quelques techniques permettant de réduire la taille de la mémoire utilisée pour le stockage des régions estimées. En effet, lorsque le modèle du système à diagnostiquer est diagnosticable par rapport à une fenêtre temporelle θ, nous pouvons éliminer toute région d'état dans

d_{cour} associée à une étiquette $\mathcal{F}_i$; $i \in \{1, \ldots, m\}$ et calculée depuis plus que θ_i u.t.. En d'autres termes, toute estimation associée à une étiquette $\mathcal{F}_i$ expire après l'écoulement de θ_i u.t. puisqu'elle ne peut pas inclure l'état courant du système, sinon, la condition de diagnosticabilité est contredite.

La complexité de la vérification de la diagnosticabilité à HTL correspond essentiellement à la complexité de la fonction d'analyse accessibilité, appliquée sur l'AHR $H^{\theta}_{i,\bar{i}}$, pour $i \in \{1, \ldots, m\}$. Afin de calculer cette complexité, nous supposons une exécution sur deux étapes de cette procédure (la même démarche considérée dans la preuve du théorème 3). Dans la première étape, l'algorithme parcourt la restriction $H^{0}_{i,\bar{i}} \otimes TB^{\theta}$ de l'AHR $H^{\theta}_{i,\bar{i}}$. D'après la propsition 4, cette restriction correspond à un AHR initialisé. Par conséquent, la complexité de l'algorithme d'analyse, appliqué sur cette restriction, est PSPACE. D'un autre coté, la complexité d'exécution de la seconde étape est exponentielle en fonction du nombre de sommets de cet automate ; i.e., elle est égale à $O(k^{\lfloor k.\frac{\theta_{max}}{d} \rfloor})$, où k désigne le nombre de sommets de $H^{\theta}_{i,\bar{i}}$, $\theta_{max} = max(\theta_1, \ldots, \theta_m)$ et d la durée minimale pour l'exécution d'un cycle dans $H^{\theta}_{i,\bar{i}}$. En conclusion, la complexité de la vérification de la diagnosticabilité à HTL est exponentielle.

5.7 Conclusion

Nous avons présenté dans ce chapitre une démarche de diagnostic pour les SDH modélisés par des automates hybrides rectangulaires vérifiant certaines conditions. Cette démarche nécessite la conception d'un modèle complet du système puis l'application d'une procédure de diagnostic en-ligne, qui estime l'état courant du système ainsi que les occurrences de défauts. L'élaboration d'une estimation peut être déclenchée par l'observation d'un événement ou le dépassement d'un délai seuil d'attente. Nous avons présenté par la suite une notion de diagnosticabilité appelée diagnosticabilité à horizon de temps limité. Cette notion permet de garantir l'identification de tout mode de défaillance au bout d'un délai fini de temps défini par une fenêtre temporelle θ. Nous avons développé une méthode systématique permettant la vérification de la diagnosticabilité à HTL. Cette méthode repose sur l'application d'une fonction d'analyse d'accessibilité sur un ensemble de produits synchrones d'automates, construits à partir du modèle du système. Enfin, nous avons présenté quelques éléments sur l'implémentation et la complexité des algorithmes utilisés dans notre travail.

Chapitre 6

Conclusion générale et perspectives

L'objectif de cette thèse était la mise en œuvre d'une démarche de diagnostic pour les systèmes à événements discrets temporisés et une sous-classe de systèmes dynamiques hybrides. Nous nous sommes basés sur un ensemble de travaux de référence (Sampath et al., 1995, 1996; Tripakis, 2002) pour l'élaboration de notre démarche.

Dans un premier temps, nous nous sommes intéressés au diagnostic des systèmes temporisés, en utilisant le modèle automate temporisé. Ce modèle permet de représenter explicitement l'information temporelle dans un SED à travers l'utilisation d'un ensemble d'horloges réelles. Ainsi, notre approche permet l'identification des défauts qui s'expriment par un changement des dates d'occurrence des événements observables. La solution présentée dans notre travail repose sur la construction hors-ligne d'un diagnostiqueur. Cette solution hors-ligne nous permet de minimiser le calcul effectué pendant la phase en-ligne du diagnostic. Le calcul en-ligne se réduit dans notre cas au franchissement d'une transition dans l'automate temporisé du diagnostiqueur et une simple évaluation de son sommet courant. Afin de contourner l'indécidabilité du problème de synthèse du diagnostiqueur à partir du modèle général d'automate temporisé, nous avons considéré un ensemble d'hypothèses sur le modèle du système à diagnostiquer, telles que l'absence d'affectations d'horloges dans les transitions sur des événements non observables et la spécification de conditions de gardes bornées dans les sommets non-finaux.

Notre démarche de diagnostic à base de modèles SED temporisés commence par la conception d'un modèle automate temporisé vérifiant certaines hypothèses. Ce modèle décrit le comportement normal et les comportements défaillants du système. Ensuite, nous avons présenté, intuitivement puis formellement, l'algorithme de synthèse du diagnostiqueur. Le diagnostiqueur obtenu suite à l'application de cet algorithme est un automate temporisé déterministe qui évolue en fonction des événements observables générés

par le système. Chaque sommet de cet automate permet de retrouver une estimation de l'état courant du système et les occurrences des événements de défauts non observables. Une fonction de décision, analyse le sommet courant du diagnostiqueur et annonce l'occurrence d'un défaut du mode F_i lorsque ce sommet est F_i-certain. Enfin, nous avons étudié la diagnosticabilité du modèle considéré dans notre démarche. Nous avons établi que l'utilisation d'un diagnostiqueur est soumise à la vérification de la notion de diagnosticabilité du modèle considéré. Une méthode systématique permettant de vérifier la diagnosticabilité du modèle automate temporisé a été élaborée. Cette méthode repose sur la détection de cycles F_i-indéterminé et des sommets finaux F_i-incertain. Un théorème liant la notion de diagnosticabilité et la vérification de ces conditions a été ensuite présenté.

Dans un second temps, nous nous sommes intéressés à l'élaboration d'une approche de diagnostic pour une sous-classe de SDH. L'élaboration de cette approche est motivée par la capacité des modèles hybrides à représenter intrinsèquement les intéractions entre la dynamique continue et la dynamique discrète d'un système réel. En plus, l'approximation du comportement d'un système en adoptant une modélisation SED temporisée peut conduire à une perte considérable d'informations nécessaires pour la discrimination des comportements défaillants. En effet, nous avons proposé une démarche de diagnostic pour une sous-classe de SHD, représentée par le modèle automate hybride rectangulaire vérifiant certaines hypothèses. Le choix de ce modèle nous a permis de contourner la complexité que pose l'analyse des modèles hybrides en général. Notre méthode de diagnostic repose sur l'utilisation d'une procédure de diagnostic en-ligne, qui estime l'état courant du système ainsi que les occurrences des défauts. L'élaboration d'une estimation peut être déclenchée par l'observation d'un événement ou le dépassement d'un délai seuil d'attente. Nous avons présenté par la suite une notion de diagnosticabilité appelée diagnosticabilité à horizon de temps limité. Cette notion permet de garantir l'identification de tout mode de défauts au bout d'un délai fini à partir du temps d'occurrence du défaut, défini par une fenêtre temporelle θ. Nous avons développé une méthode systématique permettant la vérification de la diagnosticabilité à HTL. Cette méthode repose sur l'application d'une fonction d'analyse d'accessibilité sur un ensemble de produits synchrones d'automates, construits à partir du modèle du système.

Plusieurs perspectives peuvent être envisagées, à court terme ou à long terme, comme suite au travail présenté dans cette thèse. Nous les avons regroupées selon quatre axes :

Dans un premier axe, nous envisageons implémenter les deux démarches de diagnostic développées dans ce travail. Pour ce faire, nous utiliserons les différentes bibliothèques proposées dans la littérature, permettant la modélisation et l'analyse des automates temporisés et des automates hybrides rectangulaires. Une étude de la performance des algorithmes développés sera ensuite établie afin de permettre de choisir l'implémentation la plus optimale en termes de temps d'exécution de ressources consommées.

Dans un second axe, nous visons à étendre notre démarche de diagnostic temporisé dans le cadre des SDH. Il s'agit d'élaborer une démarche de diagnostic pour les SDH basée sur la synthèse hors-ligne d'un diagnostiqueur en utilisant un modèle hybride. L'élaboration de cette démarche nécessite d'un coté, la caractérisation d'une sous-classe de modèles hybrides capable de rendre ce problème décidable et d'un autre, la conception d'un algorithme de construction du diagnostiqueur approprié à ce type de modèles.

Dans un troisième axe, nous envisageons de prendre en considération la mesurabilité de certaines variables continues au cours de notre démarche de diagnostic des SDH. L'approche de diagnostic pour les SDH présentée dans notre travail repose uniquement sur l'observation d'une trajectoire d'événements discrets temporisés générée par le système. Cependant, certaines variables telles que la température ou le débit peuvent être directement mesurées en utilisant des capteurs continus. L'abstraction de ces mesures par l'utilisation de quelques événements discrets peut conduire à des inférences erronées. En effet, une perspective de notre travail consiste à combiner notre solution avec l'approche de Lunze (Lunze, 2006) qui repose sur une fine discrétisation des trajectoires continues. Ainsi, la solution à développer doit répondre à ce compromis : minimiser la perte d'informations engendrée par la discrétisation des trajectoires continues d'un coté, et éviter le problème d'explosion combinatoire dû à cette opération d'un autre.

Dans un dernier axe, nous proposons de combiner notre approche de diagnostic à base de modèles temporisés avec la théorie du contrôle supervisé développée par Ramadge et Wonham (Ramadge et Wonham, 1987). Ce travail peut être envisagé dans le cadre d'une extension de l'approche proposée dans (Sampath et al., 1998). Il s'agit de proposer une solution intégrée qui permet de coupler le module du diagnostic avec le module du contrôle. Cette solution repose sur la synthèse d'un superviseur qui permet d'autoriser ou d'inhiber certains événements contrôllables du système afin de déterminer le plus grand sous-langage temporisé diagnosticable. Ainsi, le diagnostiqueur résultant d'une telle approche permet l'identification de tout mode de défaillance au bout d'un délai fini de l'occurrence du correspondant défaut puisque le modèle du système est toujours diagnosticable.

Annexe A

Algorithme complet de synthèse du diagnostiqueur

Nous présentons dans ce qui suit l'algorithme complet de synthèse du diagnostiqueur permettant la construction de ce dernier ainsi que son automate symbolique compagnon. La formalisation de cet algorithme nécessite l'introduction de quelques modifications sur procédure d'accessibilité non-observable $\mathcal{UR}$ pour prendre en considération la construction de l'automate compagnon. Les modifications introduites par rapport aux anciennes versions de ces deux algorithmes seront marquées en caractères gras.

Algorithm 8 Fonction d'accessibilité non-observable $\mathcal{UR}$ enrichie

Entrées : $\{[(q_i, z_i), \phi_i], i \in \{1, \ldots, k\}\}$

Initialiser $\acute{E}TATS\_ACCESSIBLES$ et $A\_TRAITER$ par $\{[(q_i, Post_t(z_i)), \phi_i],\ i \in \{1, \ldots, k\}\}$.

tantque $A\_TRAITER \neq \{\}$ **faire**

Retirer l'élément $[(q, z), \phi]$ de $A\_TRAITER$.

pour tout $q \xrightarrow{\sigma_u, g} \tilde{q} \in E, \sigma_u \in \Sigma_{uo}$ **faire**

Calculer $(\tilde{q}, \tilde{z}) := Post((q, z), \xrightarrow{\sigma_u, g})$ et $\tilde{\phi} := \phi \otimes \sigma_u$.

si $[(\tilde{q}, \tilde{z}), \tilde{\phi}] \notin \acute{E}TATS\_ACCESSIBLES$ et $\tilde{z} \neq \emptyset$ **alors**

Ajouter $[(\tilde{q}, \tilde{z}), \tilde{\phi}]$ à $A\_TRAITER$, $\acute{E}TATS\_ACCESSIBLES$.

Ajouter $[(\tilde{q}, \tilde{z}), \tilde{\phi}]$ **à** Q^s.

Ajouter $[(q, z), \phi] \xrightarrow{\sigma_u, g} [(\tilde{q}, \tilde{z}), \tilde{\phi}]$ **à** E^s.

finsi

fin pour

fin tantque

retourner $\acute{E}TATS\_ACCESSIBLES$

Algorithm 9 Procédure de synthèse du diagnostiqueur enrichi

Entrées : : Automate temporisé du système model $A = (Q, X, \Sigma, E, q_0, Q^f, I)$.

Sorties : Automate temporisé du diagnostiqueur $D = (Q^d, X, \Sigma_o, E^d, q_0^d, Q_f^d, I^d)$ **et son automate symbolique compagnon** $S = (Q^s, X, \Sigma, E^s, q_0^s, Q_f^s, I^s)$.

$q_0^d \leftarrow (\{(q_0, N)\}, z_0)$, $Q^d \leftarrow \{q_0^d\}$, $SOMMETS\_A\_TRAITER \leftarrow \{q_0^d\}$, et $E^d \leftarrow \{\}$

$Q^s \leftarrow \{(q_0, z_0, \mathcal{N})\}$, $E^s \leftarrow \emptyset$

tantque $SOMMETS\_A\_TRAITER \neq \{\}$ **faire**

retirer un sommet q^d de $SOMMETS\_A\_TRAITER$

calculer les états d'ombre de q^d, $\acute{E}TATS\_OMBRE = \mathcal{UR}(q^d)$.

déterminer $q_f^d = (\{(q_i, \phi_i), i \in \{1, \ldots, k\}\}, z_f) | (q_i, \phi_i, z_i) \in \acute{E}TATS\_OMBRE$ et $q_i \in Q_f$, $\forall i \in \{1, \cdots\}$, et $z_f = z_1 \wedge \cdots \wedge z_k)\}$

si q_f^d n'est pas vide **alors**

déterminer $\tau = min\{d \in \{1, \ldots, K+1\} | \forall (q, z, \phi) \in \acute{E}TATS\_OMBRE, (\langle z \wedge (y < d) \rangle = \emptyset) \Rightarrow q \in Q_f\}$

ajouter q_f^d à Q^d et Q_f^d, et ajouter une transition urgente $q^d \xrightarrow{\tau} q_f^d$

finsi

pour tout $\sigma \in \Sigma_o$ **faire**

déterminer l'ensemble $\acute{E}TATS\_FRANCHISSABLES^\sigma = \{(q_i, \eta_i, \phi_i) | (q_i, z_i, \phi_i) \in \acute{E}TATS\_OMBRE, q_i \xrightarrow{\sigma, g, Y} \tilde{q}_i \in E,\ and\ \eta_i = (z_i \wedge g) \neq \emptyset\}$

tantque $\exists (q_i, \eta_i, \phi_i), (q_j, \eta_j, \phi_j) \in \acute{E}TATS\_FRANCHISSABLES^\sigma \mid (\eta_i \wedge \eta_j \neq \emptyset)$ and $(\eta_i \neq \eta_j)$ **faire**

Retirer (q_i, η_i, ϕ_i) et (q_j, η_j, ϕ_j) de $\acute{E}TATS\_FRANCHISSABLES^\sigma$.

Ajouter $\{(q_i, \langle \eta_i \wedge \eta_j \rangle, \phi_i), (q_j, \langle \eta_i \wedge \eta_j \rangle, \phi_j), (q_j, \langle \neg \eta_i \wedge \eta_j \rangle, \phi_j), (q_i, \langle \eta_i \wedge \neg \eta_j \rangle, \phi_i)\}$ à $\acute{E}TATS\_FRANCHISSABLES^\sigma$.

fin tantque

$ZONES\_TRAITÉES \leftarrow \{\}$

pour tout zone $\eta \in \mathcal{C}(X) | \exists (q, \eta, \phi) \in \acute{E}TATS\_FRANCHISSABLES^\sigma$ et $\eta \notin ZONES\_TRAITÉES$ **faire**

ajouter η à $ZONES\_TRAITÉES$

créer un nouveau sommet $\tilde{q}^d = (\{(\tilde{q}_i, \phi_i), i \in \{1, \ldots, k\}\}, \tilde{z})$, où $\tilde{z} = \eta[Y \leftarrow 0]$, $(q_i, \eta, \phi_i) \in \acute{E}TATS\_FRANCHISSABLES^\sigma$, et $q_i \xrightarrow{\sigma, g, Y} \tilde{q}_i \in E$.

ajouter le sommet $\tilde{q}^d$ à Q^d et $SOMMETS\_A\_TRAITER$, si $\tilde{q}^d \notin Q^d$.

ajouter une transition $q^d \xrightarrow{\sigma, \eta, Y} \tilde{q}^d$ à E^d.

pour tout $(q_i, \eta, \phi_i) \in \boldsymbol{\acute{E}TATS\_FRANCHISSABLES^\sigma} \mid \exists\ q_i \xrightarrow{\sigma, g, Y} \tilde{q}_i \in E\ and\ \eta \subseteq z$ **faire**

ajouter $(\tilde{q}_i, \tilde{\eta}, \phi_i)$ **à** Q_s**, où** $\tilde{\eta} = \eta[Y \leftarrow 0]$.

ajouter $(q_i, \eta, \phi_i) \xrightarrow{\sigma, \eta, Y} (\tilde{q}, \theta, \eta[Y \leftarrow 0])$ **à** E_s.

fin pour

fin pour

fin pour

fin tantque

Annexe B

Preuves

B.1 Preuves du chapitre 4

Lemme 2 Soit q^d un sommet du diagnostiqueur alors :

1. pour chaque élément $(q, \phi) \in Dis(q^d)$, $v \in Z(q^d)$, il existe une exécution de A sur une trace temporisée ω, telle que $(q_0, v_0) \overset{\omega}{\rightsquigarrow} (q, v)$.
2. S'il existe une paire d'éléments $(q, \phi), (\tilde{q}, \tilde{\phi}) \in Dis(q^d)$, alors pour toute valuation $v \in Z(q^d)$, il existe une paire d'exécution de A sur traces temporisées ω et $\tilde{\omega}$, respectivement, telle que $(q_0, v_0) \overset{\omega}{\rightsquigarrow} (q, v)$, $(q_0, v_0) \overset{\tilde{\omega}}{\rightsquigarrow} (\tilde{q}, v)$ et $P(\omega) = P(\tilde{\omega})$.

Preuve :
Nous prouvons ce lemme par induction sur le nombre de transitions discrètes n dans le diagnostiqueur, effectuées pour accéder au sommet q_n^d à partir du sommet initial du diagnostiqueur q_0^d.

Base : Le sommet initial du diagnostiqueur $q_0^d = (\{(q_0, \mathcal{N})\}, z_0)$ vérifie la propriété en considérant que ω est égale à la trace vide.

induction : Nous supposons que la propriété est vraie pour l'ordre $n \geq 0$, alors pour chaque sommet q_n^d, accessible suite à n discrete transitions et pour chaque valuation $v \in Z(q_n^d)$, le lemme est vérifié. Nous avons besoin de prouver que ce lemme est vrai pour tous les sommets accessibles suite à travers $n+1$ transitions discrètes.

Soit q_{n+1}^d un sommet du diagnostiqueur accessible à travers $n+1$ transitions discrètes dans D, alors, par construction du diagnostiqueur, il existe au moins un prédécesseur de q_{n+1}^d, noté q_n^d, tel que q_n^d est accessible sur n transitions discrètes à partir du sommet initial du diagnostiqueur. Soit $q_n^d \xrightarrow{\sigma_{n+1}, \eta_{n+1}, Y_{n+1}} q_{n+1}^d$ la transition du sommet q_n^d vers le

sommet q^d_{n+1}, alors :

1)

1-a) il existe un chemin de transitions $(q_n, \phi_n, z_n) \xrightarrow{\sigma^1_{uo}, g^1_{uo}} (q^1_n, \phi^1_n, z^1_n) \xrightarrow{\sigma^2_{uo}, g^2_{uo}} \ldots \xrightarrow{\sigma^k_{uo}, g^k_{uo}} (q^k_n, \phi^k_n, z^k_n) \xrightarrow{\sigma_{n+1}, \eta_{n+1}, Y_{n+1}} (q_{n+1}, \phi_{n+1}, z_{n+1})$ dans l'automate S, où $\sigma^1_{uo}, \ldots, \sigma^k_{uo} \in \Sigma_{uo}$. Par conséquence, pour toute valuation $v_{n+1} \in Z(q^d_{n+1})$, il existe un état (q_n, v_n) où $v_n \in z_n$ et une trace ω_u telle que $(q_n, v_n) \overset{\omega_u}{\rightsquigarrow} (q_{n+1}, v_{n+1})$.

1-b) A partir de notre hypothèse d'induction sur q^d_n, il existe une trace $\omega_n \in L$, telle que $(q_0, v_0) \overset{\omega_n}{\rightsquigarrow} (q_n, v_n)$.

Nous concluons à travers (a) et (b) qu'il existe une trace de L, $\omega_{n+1} = '\omega_n.\omega_u'$, telle que $(q_0, v_0) \overset{\omega_{n+1}}{\rightsquigarrow} (q_{n+1}, v_{n+1})$.

2) Maintenant, nous supposons que q^d_{n+1} contient au moins une paire d'éléments $(q_{n+1}, \phi_{n+1}), (\tilde{q}_{n+1}, \tilde{\phi}_{n+1}) \in Dis(q^d_{n+1})$. Soit $v_{n+1} \in Z_{n+1}$, nous allons prouver qu'il existe une paire d'exécution sur les traces $\omega_{n+1}, \tilde{\omega}_{n+1} \in L$, respectivement, atteignant les états (q_{n+1}, v_{n+1}) et $(\tilde{q}_{n+1}, v_{n+1})$ avec $P(\omega_{n+1}) = P(\tilde{\omega}_{n+1})$.

2-a) Par construction du diagnostiqueur, il existe deux chemins de transitions dans S, $(q_n, \phi_n, z_n) \xrightarrow{\sigma^1_{uo}, g^1_{uo}} (q^1_n, \phi^1_n, z^1_n) \xrightarrow{\sigma^2_{uo}, g^2_{uo}} \ldots \xrightarrow{\sigma^k_{uo}, g^k_{uo}} (q^k_n, \phi^k_n, z^k_n) \xrightarrow{\sigma_{n+1}, \eta_{n+1}, Y_{n+1}} (q_{n+1}, \phi_{n+1}, z_{n+1})$ et $(\tilde{q}_n, \tilde{\phi}_n, z_n) \xrightarrow{\tilde{\sigma}^1_{uo}, \tilde{g}^1_{uo}} (\tilde{q}^1_n, \tilde{\phi}^1_n, \tilde{z}^1_n) \xrightarrow{\tilde{\sigma}^2_{uo}, \tilde{g}^2_{uo}} \ldots \xrightarrow{\tilde{\sigma}^{k'}_{uo}, \tilde{g}^{k'}_{uo}} (\tilde{q}^{k'}_n, \tilde{\phi}^{k'}_n, \tilde{z}^{k'}_n) \xrightarrow{\sigma_{n+1}, \eta_{n+1}, Y_{n+1}} (\tilde{q}_{n+1}, \tilde{\phi}_{n+1}, z_{n+1})$, tels que $\sigma^1_{uo}, \ldots, \sigma^k_{uo}, \tilde{\sigma}^1_{uo}, \ldots, \tilde{\sigma}^{k'}_{uo} \in \Sigma_{uo}$.

Par construction du prédicat η_{n+1}, nous avons $\eta_{n+1} \subseteq z^k_n$ et $\eta_{n+1} \subseteq \tilde{z}^{k'}_n$. Soient (q^k_n, v^e_{n+1}) et $(\tilde{q}^k_n, v^e_{n+1})$ deux états tels que $v^e_{n+1} \in \eta_{n+1}$ et $v_{n+1} = v^e_{n+1}[Y_{n+1} \leftarrow 0]$. Alors, il existe une paire d'exécution sur deux traces temporisées ω_u et ω'_u (dans $(\Sigma_{uo} \times \mathbb{R}_+)^*$), et une paire d'états (q_n, v_n) et $(\tilde{q}_n, \tilde{v}_n)$, où $v_n \in z_n$ et $\tilde{v}_n \in \tilde{z}_n$, telle que $(q_n, v_n) \overset{\omega_u}{\rightsquigarrow} (q^k_n, v^e_{n+1})$ et $(\tilde{q}_n, v_n) \overset{\omega'_u}{\rightsquigarrow} (\tilde{q}^k_n, v^e_{n+1})$. Puisque les transitions sur les événements non observables ne permettent pas les affectations d'horloges (Hypothèse $\mathcal{H}_2$), il existe $d, \tilde{d} \in \mathbb{R}_+$, tels que $v^e_{n+1} = v_n + d$ et $v^e_{n+1} = \tilde{v}_n + \tilde{d}$. Par conséquence, $\tilde{v}_n = v_n + (d - \tilde{d})$ (nous supposons que $d - \tilde{d} \geq 0$, sinon, nous inversons les termes de l'égalité). Par conséquence, l'état $(\tilde{q}_n, \tilde{v}_n)$ est accessible depuis l'état $(\tilde{q}_n, v_n)$ suite à l'écoulement de $d - \tilde{d}$ u.t. ; i.e., $(\tilde{q}_n, v_n) \xrightarrow{d-\tilde{d}} (\tilde{q}_n, \tilde{v}_n)$. Nous notons que cette transition est possible puisque l'invariant du sommet $\tilde{q}_n$ est convexe et égale à l'invariant du sommet q_n. Par conséquence, il existe une exécution sur la trace temporisée $\tilde{\omega}_u$, telle que $(\tilde{q}_n, v_n) \overset{\tilde{\omega}_u}{\rightsquigarrow} (\tilde{q}^k_n, v^e_{n+1})$, et $\tilde{\omega}_u = '(d - \tilde{d}).\omega'_u'$. Il est clair que $temps(\omega_u) = temps(\tilde{\omega}_u) = d$.

2-b) Selon notre hypothèse d'induction, il existe une paire de traces $\omega_n, \tilde{\omega}_n \in L$, telle que $(q_0, v_0) \overset{\omega_n}{\rightsquigarrow} (q_n, v_n)$, $(q_0, v_0) \overset{\tilde{\omega}_n}{\rightsquigarrow} (\tilde{q}_n, v_n)$ et $P(\omega) = P(\tilde{\omega})$.

En concussion, d'après (a) et (b), il existe une paire de traces telle que $(q_0, v_0) \overset{\omega_{n+1}}{\rightsquigarrow} (q_{n+1}, v_{n+1})$ et $(q_0, v_0) \overset{\tilde{\omega}_{n+1}}{\rightsquigarrow} (\tilde{q}_{n+1}, v_{n+1})$, où ω_{n+1} = '$\omega_n.\omega_u.\sigma_{n+1}$' et $\tilde{\omega}_{n+1}$ = '$\tilde{\omega}_n.\tilde{\omega}_u.\sigma_{n+1}$'. En plus, $P(\omega_{n+1})$='$P(\omega_n).temps(\omega_u).\sigma_{n+1}$'. Puisque $temps(\omega_u) = temps(\tilde{\omega}_u)$ et $P(\omega) = P(\tilde{\omega})$, alors, $P(\omega_{n+1}) = P(\tilde{\omega}_{n+1})$.

□

Lemme 5

Soit q^d_{n+1} un sommet d'un cycle F_i-indéterminé et soit $(q_{n+1}, \mathcal{F}_i), (\tilde{q}_{n+1}, \tilde{\phi}_{n+1}) \in Dis(q^d_{n+1})$, tel que $\tilde{\phi}_{n+1} \neq \mathcal{F}_i$, $(q_{n+1}, \mathcal{F}_i, z_{n+1}) \in \mathcal{C}_i$ et $(\tilde{q}_{n+1}, \tilde{\phi}_{n+1}, z_{n+1}) \in \overline{\mathcal{C}_i}$, où $z_{n+1} = Z(q^d_{n+1})$. Nous notons par q^d_n le prédécesseur de q^d_{n+1} dans le cycle F_i-indéterminé, sur la transition $q^d_n \xrightarrow{\sigma_{n+1}, g_{n+1}, Y_{n+1}} q^d_{n+1} \in E^d$. Alors, pour toute valuation $v_{n+1} \in z_{n+1}$, il existe une valuation $v_n \in z_n = Z(q^d_n)$, une paire d'éléments $(q_n, \mathcal{F}_i), (\tilde{q}_n, \tilde{\phi}_n) \in Dis(q^d_n)$ et une paire de traces temporisées ω_{n+1} et $\tilde{\omega}_{n+1}$, telles que :

- $(q_n, \mathcal{F}_i, z_n) \in \mathcal{C}_i$ et $(\tilde{q}_n, \tilde{\phi}_n, z_n) \in \overline{\mathcal{C}_i}$;
- $(q_n, v_n) \overset{\omega_{n+1}}{\rightsquigarrow} (q_{n+1}, v_{n+1})$ et $(q_n, v_n) \overset{\tilde{\omega}_{n+1}}{\rightsquigarrow} (q_{n+1}, v_{n+1})$;
- $P(\omega_{n+1}) = P(\tilde{\omega}_{n+1})$ = '$d.\sigma_{n+1}$', où $d \in \mathbb{R}_+$.

Preuve : Nous allons utiliser la technique employée pour la preuve du Lemme 2.

Par construction du diagnostiqueur, il existe deux chemins de transitions dans S, $(q_n, \mathcal{F}_i, z_n) \xrightarrow{\sigma^1_{uo}, g^1_{uo}} (q^1_n, \mathcal{F}_i, z^1_n) \xrightarrow{\sigma^2_{uo}, g^2_{uo}} \dots \xrightarrow{\sigma^k_{uo}, g^k_{uo}} (q^k_n, \mathcal{F}_i, z^k_n) \xrightarrow{\sigma_{n+1}, \eta_{n+1}, Y_{n+1}} (q_{n+1}, \mathcal{F}_i, z_{n+1})$ dans $\mathcal{C}_i$ et $(\tilde{q}_n, \tilde{\phi}_n, z_n) \xrightarrow{\tilde{\sigma}^1_{uo}, \tilde{g}^1_{uo}} (\tilde{q}^1_n, \tilde{\phi}^1_n, \tilde{z}^1_n) \xrightarrow{\tilde{\sigma}^2_{uo}, \tilde{g}^2_{uo}} \dots \xrightarrow{\tilde{\sigma}^{k'}_{uo}, \tilde{g}^{k'}_{uo}} (\tilde{q}^{k'}_n, \tilde{\phi}^{k'}_n, \tilde{z}^{k'}_n) \xrightarrow{\sigma_{n+1}, \eta_{n+1}, Y_{n+1}} (\tilde{q}_{n+1}, \tilde{\phi}_{n+1}, z_{n+1})$ dans $\overline{\mathcal{C}_i}$, tel que $\tilde{\phi}_n, \tilde{\phi}^1_n, \dots, \tilde{\phi}^{k'}_n, \tilde{\phi}_{n+1} \neq \mathcal{F}_i$.

Par construction du prédicat η_{n+1}, nous avons $\eta_{n+1} \subseteq z^k_n$ et $\eta_{n+1} \subseteq \tilde{z}^{k'}_n$. Soient (q^k_n, v^e_{n+1}) et $(\tilde{q}^{k'}_n, v^e_{n+1})$ deux états tels que $v^e_{n+1} \in \eta_{n+1}$ et $v_{n+1} = v^e_{n+1}[Y_{n+1} \leftarrow 0]$. Alors, il existe une paire de traces temporisées $\omega_u, \omega'_u \in (\Sigma_{uo} \times \mathbb{R}_+)^*$ et une paire d'états, (q_n, v_n) et$(\tilde{q}_n, \tilde{v}_n)$, où $v_n \in z_n$, $\tilde{v}_n \in \tilde{z}_n$, $(q_n, v_n) \overset{\omega_u}{\rightsquigarrow} (q^k_n, v^e_{n+1})$ et $(\tilde{q}_n, v_n) \overset{\omega'_u}{\rightsquigarrow} (\tilde{q}^{k'}_n, v^e_{n+1})$.

Puisque les transitions sur les événements non observables ne réinitialisent pas les horloges, (hypothèse $\mathcal{H}_2$), alors il existe $d, \tilde{d} \in \mathbb{R}_+$ tel que $v^e_{n+1} = v_n + d$ et $v^e_{n+1} = \tilde{v}_n + \tilde{d}$. Par conséquence, $\tilde{v}_n = v_n + (d - \tilde{d})$ (nous supposons que $d - \tilde{d} \geq 0$, sinon, nous inversons les termes de l'égalité).

L'état $(\tilde{q}_n, \tilde{v}_n)$ est accessible depuis $(\tilde{q}_n, v_n)$ sur l'écoulement de $d - \tilde{d}$ u.t. ; i.e.,$(\tilde{q}_n, v_n) \xrightarrow{d - \tilde{d}} (\tilde{q}_n, \tilde{v}_n)$. Par conséquence, $(\tilde{q}_n, v_n) \overset{\tilde{\omega}_u}{\rightsquigarrow} (\tilde{q}^k_n, v^e_{n+1})$, où $\tilde{\omega}_u$ = '$(d - \tilde{d}).\omega'_u$'. En plus, il est clair que $temps(\omega_u) = temps(\tilde{\omega}_u)$.

Soient ω_{n+1} ='$\omega_u.\sigma_{n+1}$' et $\tilde{\omega}_{n+1}$ = '$\tilde{\omega}_u.\sigma_{n+1}$'. Il est clair que $(q_n, v_n) \overset{\omega_{n+1}}{\rightsquigarrow} (q_{n+1}, v_{n+1})$ et que $(\tilde{q}_n, v_n) \overset{\tilde{\omega}_{n+1}}{\rightsquigarrow} (\tilde{q}_{n+1}, v_{n+1})$. En plus, $P(\omega_{n+1}) = P(\tilde{\omega}_{n+1})$='$temps(\omega_u).\sigma_{n+1}$'.

□

Lemme 6
Soit q_n^d un sommet du diagnostiqueur appartenant à un cycle F_i-indéterminé et $(q_n, \mathcal{F}_i), (\tilde{q}_n, \tilde{\phi}_n) \in Dis(q^d)$, tels que $\tilde{\phi}_n \neq \mathcal{F}_i$, $(q_n^p, \mathcal{F}_i, z_n) \in \mathcal{C}_i$, $(\tilde{q}_n^q, \tilde{\phi}_n^q, z_n) \in \overline{\mathcal{C}_i}$ et $z_n = Z(q_n^d)$.

Alors, étant donné un entier naturel $\alpha \geq 0$ et $\forall v_{n,p}^0 \in z_n$, il existe une valuation d'horloges $v_{n,p}^\alpha \in z_n$ et une paire de traces temporisées $\omega^\alpha, \tilde{\omega}^\alpha$ telles que :

1. L'exécution $(q_n^p, v_{n,p}^\alpha) \overset{\omega^\alpha}{\rightsquigarrow} (q_n^p, v_{n,p}^0)$ fait évoluer S α fois dans le cycle $\mathcal{C}_i$,
2. L'exécution $(\tilde{q}_n^{q'}, v_{n,p}^\alpha) \overset{\tilde{\omega}^\alpha}{\rightsquigarrow} (\tilde{q}_n^q, v_{n,p}^0)$ fait évoluer S dans le cycle $\overline{\mathcal{C}_i}$,
3. $P(\omega^\alpha) = P(\tilde{\omega}^\alpha)$.

Preuve :

Nous prouvons ce lemme par induction sur le nombre de cycles α, effectuée par une exécution de S dans $\mathcal{C}_i$.

Base : $\alpha = 0$, cas trivial. ω^0 et $\tilde{\omega}^0$ sont égales à la trace vide.

Induction : nous supposons que la propriété de l'induction est vérifiée pour l'ordre order α, nous montrons qu'elle est aussi vérifiée pour l'ordre $\alpha + 1$.

D'après l'hypothèse de l'induction, il existe une valuation $v_{n,p}^\alpha \in z_n$ et une paire de traces temporisées $\omega^\alpha, \tilde{\omega}^\alpha$ telles que l'exécution $(q_n^p, v_{n,p}^\alpha) \overset{\omega^\alpha}{\rightsquigarrow} (q_n^p, v_{n,p}^0)$ fait évoluer S, pour α fois, dans le cycle $\mathcal{C}_i$, l'exécution $(\tilde{q}_n^{q'}, v_{n,p}^\alpha) \overset{\tilde{\omega}^\alpha}{\rightsquigarrow} (\tilde{q}_n^q, v_{n,p}^0)$ fait évoluer S dans le cycle $\overline{\mathcal{C}_i}$ et $P(\omega^\alpha) = P(\tilde{\omega}^\alpha)$.

Nous supposons que lorsque le système effectue un cycle d'exécution dans $\mathcal{C}_i$, il effectue au même temps p cycles dans le diagnostiqueur $q_1^d \xrightarrow{\sigma_1, g_1, Y_1} q_2^d \dots q_{n-1}^d \xrightarrow{\sigma_{n-1}, g_{n-1}, Y_{n-1}} q_n^d \xrightarrow{\sigma_n, g_n, Y_n} q_1^d$, $p \geq 1$.

Soit $v_{k+1,j}^\alpha$ une valuation d'horloges dans $Z(q_{k+1}^d)$, où $j \in \{1, \dots, p\}$ and $k \in \{1, \dots, n\}$, $q_{n+1}^d = q_1^d$.

Selon le Lemme 5, il existe $v^{\alpha}_{k,j} \in z_k$, une paire d'éléments $(q^j_k, \mathcal{F}_i), (\tilde{q}^j_k, \mathcal{F}_i) \in Dis(q^d_k$ et une paire de traces ω^j_{k+1} et $\tilde{\omega}^j_{k+1}$, telles que :

1. $(q^j_k, \mathcal{F}_i, z_k) \in \mathcal{C}_i$ et $(\tilde{q}^j_k, \tilde{\phi}_k, z_k) \in \overline{\mathcal{C}_i}$;
2. $(q^j_k, v^{\alpha}_{k,j}) \overset{\omega^j_{k+1}}{\rightsquigarrow} (q^j_{k+1}, v^{\alpha}_{k+1,j})$ et $(\tilde{q}^j_k, v^{\alpha}_{k,j}) \overset{\tilde{\omega}^j_{k+1}}{\rightsquigarrow} (\tilde{q}^j_{k+1}, v^{\alpha}_{k+1,j})$;
3. $P(\omega^j_{k+1}) = P(\tilde{\omega}^j_{k+1}) = 'd^j_{k+1}.\sigma_{k+1}'$, où $d^j_{k+1} \in \mathbb{R}_+$.

Par la construction des traces ω^p_n, $\tilde{\omega}^p_n$, ω^p_{n-1},$\tilde{\omega}^p_{n-1}$,..., ω^p_1,$\tilde{\omega}^p_1$, ω^{p-1}_n, $\tilde{\omega}^{p-1}_n$ ..., ω^1_2, $\tilde{\omega}^1_2$,ω^1_1, $\tilde{\omega}^1_1$ à partir des valuations $v^{\alpha}_{n,p}$, nous concluons qu'il existe une valuation $v^{\alpha+1}_{n,p}$ et une exécution $(q^p_n, v^{\alpha+1}_{n,p}) \overset{\omega^1_1}{\rightsquigarrow} (q^1_1, v^{\alpha}_{1,1}) \overset{\omega^1_2}{\rightsquigarrow} (q^1_2, v^{\alpha}_{2,1}) \ldots (q^1_n, v^{\alpha}_{n,1}) \overset{\omega^p_{n-1}}{\rightsquigarrow} (q^p_{n-1}, v^{\alpha}_{n-1,p}) \ldots \overset{\omega^p_n}{\rightsquigarrow} (q^p_n, v^{\alpha}_{n,p})$, faisant évoluer S dans le cycle $\mathcal{C}_i$.

Soit $\omega = '\omega^1_1.\omega^1_2.\ldots.\omega^1_n.\omega^2_1.\ldots.\omega^p_n'$ et $\tilde{\omega} = '\tilde{\omega}^1_1.\tilde{\omega}^1_1.\ldots.\tilde{\omega}^1_n.\tilde{\omega}^2_1.\ldots.\tilde{\omega}^p_n'$. Il est clair que $P(\omega) = P(\tilde{\omega})$ et que $(q^p_n, v^{\alpha+1}_{n,p}) \overset{\omega}{\rightsquigarrow} (q^p_n, v^{\alpha}_{n,p})$, et $(\tilde{q}^{q''}_n, v^{\alpha+1}_{n,p}) \overset{\omega}{\rightsquigarrow} (\tilde{q}^{q'}_n, v^{\alpha}_{n,p})$.

Soient $\omega^{\alpha+1} = '\omega.\omega^{\alpha}'$ et $\tilde{\omega}^{\alpha+1} = '\tilde{\omega}.\tilde{\omega}^{\alpha}'$. Il est clair que $P(\omega^{\alpha+1}) = P(\tilde{\omega}^{\alpha+1})$. En plus, $(q^p_n, v^{\alpha+1}_{n,p}) \overset{\omega^{\alpha+1}}{\rightsquigarrow} (q^p_n, v^0_{n,p})$ fait évoluer S, $\alpha + 1$ fois, dans le cycle $\mathcal{C}_i$ et l'exécution $(\tilde{q}^{q''}_n, v^{\alpha+1}_{n,p}) \overset{\tilde{\omega}^{\alpha+1}}{\rightsquigarrow} (\tilde{q}^q_n, v^0_{n,p})$ fait évoluer S dans le cycle $\overline{\mathcal{C}_i}$.

□

B.2 Preuves du chapitre 5

Lemme 9. (1) et (2) sont équivalents :

1. Il existe une exécution de $H_{i,\bar{i}}$, $i \in \{1, \ldots, m\}$, sur une trace temporisée ω, telle que ω contient un défaut de l'ensemble F_k.
2. Il existe un paire de traces temporisées $\omega_1 \in L_k$ et $\omega_2 \in L - L_k$, $i \in \{1, \ldots, m\}$, telle que $P(\omega_1) = P(\omega_2)$.

Preuve :

(1) $\Rightarrow$ (2) : Nous supposons qu'il existe une exécution de $H_{i,\bar{i}}$, sur une trace temporisée ω, qui atteint l'état $((q_1, q_2), \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} \end{bmatrix})$, telle que ω contient un défaut de l'ensemble F_k. D'après la proposition 3, il existe une paire d'exécutions de H_i et $H_{\bar{i}}$, respectivement sur les traces temporisées ω_1 et ω_2, menant aux états $(q_1, \mathbf{v_1})$ et $(q_2, \mathbf{v_2})$, telles que $P(\omega_1) = P(\omega_2)$. D'après la proposition 2, nous avons $\omega_1 \in L_0 \cup L_i$ et $\omega_2 \in L - L_i$.[1].

[1]Puisque chaque événement non observable $\sigma^i_k \in \Sigma_i$, existant dans la trace ω_1, correspond à un événement σ_k dans Σ, nous supposons un passage implicite entre ces deux événements pour que l'inclusion $\omega_1 \in L_0 \cup L_i$ soit correcte syntaxiquement.

Si la trace temporisée ω contient un événement de défaut de l'ensemble F_k, alors $\mathbf{v}(y^k) > 0$ pour tout $k \in \{1, \ldots, m\}$. Puisque la valuation d'horloges v correspond l'union des valuations $\mathbf{v_1}$ et $\mathbf{v_2}$, alors, selon la valeur de k ($k = i$ ou $k \in \{1, \ldots, m\} - \{i\}$), nous avons $\mathbf{v_1}(y^k) > 0$ ou $\mathbf{v_2}(y^k) > 0$. En effet, nous distinguons deux cas :

- $\mathbf{k{=}i}$: Le chronomètre y_i est défini dans l'AHR H_i et $\mathbf{v_1}(y^i) > 0$. Par conséquence, ω_1 contient un défaut de l'ensemble F_i ; i.e., $\omega_1 \in L_i$. En conclusion, nous avons $\omega_1 \in L_i$ et $\omega_2 \in L - L_i$.
- $\mathbf{k} \in \{\mathbf{1}, \ldots, \mathbf{m}\} - \{\mathbf{i}\}$: Le chronomètre y^k est défini dans l'AHR $H_{\bar{i}}$, et par conséquence, $\mathbf{v_2}(y^k) > 0$. D'un coté, ω_2 contient un défaut de l'ensemble F_k ; i.e., $\omega_2 \in L_k$. D'un autre, $\omega_1 \in L_0 \cup L_i$ et d'après la proposition 2, $L_0 \subset L - L_k$ et $L_i \subset L - L_k$. En conclusion, nous avons $\omega_1 \in L - L_k$.

□

(2) $\Rightarrow$ (1) : Dans le sens inverse, nous supposons qu'il existe une paire de traces $\omega_1 \in L_k$ et $\omega_2 \in L - L_k$, telle que $P(\omega_1) = P(\omega_2)$. En effet, $\omega_1 \in L(H_k)$ et $\omega_2 \in L(H_{\overline{k}})$. D'après la proposition 3, il existe une exécution de $H_{k,\overline{k}}$ sur une trace temporisée ω qui correspond à la synchronisation des exécutions de H_k et $H_{\overline{k}}$, sur respectivement, ω_1 et ω_2. Puisque ω_1 contient un défaut de l'ensemble F_k et $H_{k,\overline{k}} = H_k \otimes H_{\overline{k}}$, alors, la trace ω contient une défaut de l'ensemble F_k.

Théorème 4. La fonction d'analyse d'accessibilité, appliquée aux AHRs $H^{\theta}_{i,\bar{i}}$, pour tout $i \in \{1, \ldots, m\}$, s'arrête au bout d'un temps fini.

Preuve : Afin de prouver l'arrêt de cet algorithme, nous divisons son exécution sur deux étapes : (1) dans la première étape, l'algorithme explore la restriction $H^0_{i,\bar{i}} \otimes TB$ de $H^{\theta}_{i,\bar{i}}$, $i \in \{1, \ldots, m\}$. Cette partie de l'algorithme suspend le calcul du successeur de chaque état symbolique atteint suite à une transition sur un événement de défaut. Nous plaçons cet état symbolique dans un ensemble intermédiaire, appelé $EN\_ATTENTE$, pour qu'il soit traité ultérieurement.

(2) Dans la seconde étape de l'exécution de la fonction, nous initialisons l'ensemble $A\_TRAITER$ par le contenu de l'ensemble $EN\_ATTENTE$, puis nous exécutions l'algorithme pour explorer les successeurs des états symboliques dans $EN\_ATTENTE$, qui n'ont pas été explorés lors de la première étape. Nous pouvons établir, facilement, que l'exécution de ces deux étapes est équivalente à une exécution de la fonction.

La l'arrêt de l'exécution de la première étape est garantie. Les successeurs des états symboliques sur des transitions de défauts ne vont pas être explorés. Par conséquence, l'algorithme va effectuer l'analyse d'accessibilité de la restriction $H^0_{i,\bar{i}} \otimes TB^{\theta}$ de $H^{\theta}_{i,\bar{i}}$. Puisque $H^0_{i,\bar{i}}$ est un AHR initialisé, (d'après la proposition 4), cette première étape

d'exécution de la fonction d'analyse d'accessibilité s'arrête au bout d'un nombre fini d'itérations (Henzinger et al., 1998).

Dans la seconde étape, nous exécutons l'algorithme à partir des états symboliques de l'ensemble $EN\_ATTENTE$. Chaque état dans l'ensemble $EN\_ATTENTE$ a été atteint par une transition sur un événement de défaut. D'après le lemme 1, $H^{\theta}_{i,\bar{i}}$ est fortement non-zénon. En plus, pour chaque état symbolique dans $EN\_ATTENTE$, il existe un chronomètre $y^k, k \in \{1,\ldots,m\}$ qui sera actif; i.e.,$\dot{y}^k = 1$, dans tous les successeurs des états considérés. En effectuant le calcul en avant des successeurs, la propriété FNZ assure que chaque chronomètre y^k est incrémenté après chaque cycle du système par une valeur uniformément bornée par d jusqu'à atteindre la valeur θ_k. Par conséquence, le chronomètre y^k atteint la valeur θ_k au bout d'un nombre fini d'itérations et ainsi, un sommet marqué sera accessible. Nous notons qu'un blocage de temps dans l'exécution en cours se produit lorsque $y^k = \theta_k$, après avoir atteint un sommet marqué. Ceci permet de garantir l'arrêt de la fonction d'analyse d'accessibilité pour les traces contenant des défauts.

□

Lemme 10. Les propositions suivantes sont équivalentes :

1. Il existe un sommet marqué dans l'AHR $H^{\theta}_{i,\bar{i}}$, $i \in \{1,\ldots,m\}$, accessible depuis un état initial.
2. Il existe une paire de traces temporisées $\omega_1 \in L_k^{\theta_k}$ et $\omega_2 \in L - L_k$, $k \in \{1,\ldots,m\}$, telle que $P(\omega_1) = P(\omega_2)$.

Preuve :

(1) $\Rightarrow$ (2) : Nous supposons qu'il existe $i \in \{1,\ldots,m\}$ et une exécution sur la trace $\widehat{\omega}$ dans $H^{\theta}_{i,\bar{i}}$, atteignant l'état $(\widehat{q}, \mathbf{v})$, où $\widehat{q} \in M_{i,\bar{i}}$ et $\mathbf{v}$ est une valuation de $X_{i,\bar{i}}$. Puisque chaque transition vers un sommet marqué dans $H^{\theta}_{i,\bar{i}}$ se produit sur l'événement μ, alors, la trace $\widehat{\omega}$ admet comme suffixe l'événement μ. Puisque toutes les conditions de garde dans les transitions de $H^{\theta}_{i,\bar{i}}$ ont la forme $y^k \geq \theta_k$, $k \in \{1,\ldots,m\}$,alors il existe $k \in \{1,\ldots,m\}$, tel que $\mathbf{v}(y^k) \geq \theta_k$. Par conséquent, la trace $\widehat{\omega}$ contient nécessairement un défaut de l'ensemble F_k.

Puisque $H^{\theta}_{i,\bar{i}} = H_{i,\bar{i}} \otimes TB^{\theta}$, nous supposons que le sommet $\widehat{q}$ correspond à une paire $(q_{i,\bar{i}}, q_M)$, où $q_{i,\bar{i}}$ est un sommet de l'AHR $H_{i,\bar{i}}$. Soit ω une trace temporisée dont l'exécution atteint l'état $(q_{i,\bar{i}}, \mathbf{v})$ (la trace ω est obtenue en éliminant l'événement μ de $\widehat{\omega}$). Puisque ω contient un défaut de l'ensemble F_k, d'après le lemme 7, il existe une paire de traces $\omega_1 \in L_k$ et $\omega_2 \in L - L_k$, telle que $P(\omega_1) = P(\omega_2)$, et atteignant respectivement, les états $(q_1, \mathbf{v_1})$ dans H_i et $(q_2, \mathbf{v_2})$ dans $H_{\bar{i}}$, où $q_{i,\bar{i}} = (q_1, q_2)$ et $\mathbf{v} = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} \end{bmatrix}$. Deux cas peuvent se présenter :

- $\mathbf{k}$=$\mathbf{i}$, ω_1 contient un défaut de F_i et $\mathbf{v_1}(y^i) > \theta_i$. Par conséquent, $\omega_1 \in L_i^{\theta_i}$ et $\omega_2 \in L - L_i$.
- $\mathbf{k} \in \{\mathbf{1}, \ldots, \mathbf{m}\} - \{\mathbf{i}\}$, ω_2 contient un défaut de l'ensemble $F_k, k \neq i$ et $\mathbf{v_2}(y^k) > \theta_k$. Par conséquent, $\omega_1 \in L - L_k$ et $\omega_2 \in L_k^{\theta_k}$.

En conclusion, il existe, dans les deux cas, une paire de traces dans $L_k^{\theta_k}$ et $L - L_k$, $k \in \{1, \ldots, m\}$, ayant des projections observables identiques.

(2) $\Rightarrow$ (1) :

Nous supposons qu'il existe $k \in \{1, \ldots, m\}$ et deux traces $\omega_1 \in L_k^{\theta_k}$ et $\omega_2 \in L - L_k$, telles que $P(\omega_1) = P(\omega_2)$. Puisque $L_k^{\theta_k} \subseteq L_k$, d'après le lemme 7, il existe $i \in \{1, \ldots, m\}$ et une trace temporisée ω, acceptée par l'AHR $H_{i,\bar{i}}$, telle que ω contient un défaut de l'ensemble F_k. Deux cas peuvent se présenter :

- $\mathbf{k} \in \{\mathbf{1}, \ldots, \mathbf{m}\} - \{\mathbf{i}\}$ et $\omega_\mathbf{2} \in \mathbf{L_i^{\theta_i}}$. Alors, ω contient au moins deux événements de défauts, respectivement des ensembles F_i et F_k. Nous notons par τ, la valeur minimale entre $(t_i + \theta_i)$ et $(t_k + \theta_k)$, où t_i et t_k désignent les dates des occurrences d'événements de F_i et F_k, respectivement, dans la trace ω. Nous extrairons le préfixe $\widehat{\omega}$ de ω, où $temps(\widehat{\omega}) = \tau$. Nous ajoutons l'événement μ comme un suffixe la trace $\widehat{\omega}$. La trace $\widehat{\omega}$ est ainsi acceptée par l'AHR $H_{i,\bar{i}}^{\theta}$, et son exécution sur cette trace atteint un sommet marqué.
- $\mathbf{k} = \mathbf{i}$ ou $\omega_\mathbf{2} \notin \mathbf{L_i^{\theta_i}}$. Nous extrairons le préfixe $\widehat{\omega}$ de ω, où $temps(\widehat{\omega}) = t_k + \theta_k$ et t_k désigne la date de la première occurrence d'un défaut de l'ensemble F_k dans ω. Nous ajoutons l'événement μ comme un préfixe de la trace $\widehat{\omega}$. L'exécution de $H_{i,\bar{i}}^{\theta}$ sur la trace $\widehat{\omega}$ atteint un sommet marqué.

Dans les deux cas, nous concluons qu'il existe une trace temporisée $\widehat{\omega}$, atteignant un sommet marqué $H_{i,\bar{i}}^{\theta}$.

□

Bibliographie

Alla, H. et R. David. 1998, «Continuous and hybrid petri nets», *Journal of Circuits, Systems and Computer*, vol. 8, n° 1, p. 159–188.

Allaham, A. 2008, *Surveillance des systèmes à événements discrets commandés : Conception et implémentation en utilisant l'automate programmable industriel*, thèse de doctorat, L'université Joseph Fourier, Grenoble.

Allam, M. et H. Alla. 1996, «From hybrid petri nets to hybrid automata», *Journal européen des systèmes automatisés*, vol. 320, n° 9-10, p. 1165–1185.

Alur, R. 1999, «Timed automata», *Theoretical Computer Science*, vol. 126, p. 183–235.

Alur, R., C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. h. Ho, X. Nicollin, A. Olivero, J. Sifakis et S. Yovine. 1995, «The algorithmic analysis of hybrid systems», *Theoretical Computer Science*, vol. 138, p. 3–34.

Alur, R., C. Courcoubetis, T. A. Henzinger et P.-H. Ho. 1993, «Hybrid automata : An algorithmic approach to the specification and verification of hybrid systems», *Hybrid Systems, LNCS*, p. 209–229.

Alur, R. et D. L. Dill. 1994, «A theory of timed automata», *Theoretical Computer Science*, vol. 126, p. 183–235.

Alur, R., L. Fix et T. A. Henzinger. 1999, «Event-clock automata : A determinizable class of timed automata», *Theoretical Computer Science*, vol. 211, p. 1–13.

Bagnara, R., E. Ricci, E. Zaffanella et P. Hill. 2002, «Possibly not closed convex polyhedra and the parma polyhedra library», dans *Int. Symp. Volume 2477 of LNCS., Springer*, p. 213–229.

Balluchi, A., L. Benvenuti, M. D. D. Benedetto et A. L. Sangiovanni-vincentelli. 2002, «Design of observers for hybrid systems», dans *Proceedings of Hybrid Systems : Computation and Control, volume 2289 of LNCS*, Springer-Verlag, p. 76–89.

Barbuti, R. et L. Tesei. 2004, «Timed automata with urgent transitions», *Acta Inf.*, vol. 40, n° 5, p. 317–347.

Basseville, M. 1988, «Detecting changes in signals and systems - a survey», *Automatica*, vol. 24, n° 3, p. 309–326.

Beard, R. V. 1971, *Failure accommodation in linear systems through self reorganization*, thèse de doctorat, Massachusetts Institute of Technology.

Ben Hadj-Alouane, N., A. Ben Hadj-Alouane, F. Lin et M. Yeddes. 2006, «A mixed integer dynamic programming approach to a class of optimal control problems in hybrid systems», *Cybernetics and Systems*, vol. 37, n° 5, p. 481–504.

Ben Hadj-Alouane, N., S. Lafortune et F. Lin. 1994, «Variable lookahead supervisory control with state information», *IEEE Transactions on Automatic Control*, vol. 39, n° 12, p. 2398–2410.

Bengtsson, J. et W. Yi. 2004, «Timed automata : Semantics, algorithms and tools», *Technical Report 316, UNU-IIST, Macau.*

Berthomieu, B. et M. Diaz. 1991, «Modeling and verification of time dependent systems using time petri nets», *IEEE Transactions on Software Engineering*, vol. 17, n° 3, p. 259–273.

Bhowal, P., D. Sarkar, S. Mukhopadhyay et A. Basu. 2007, «Fault diagnosis in discrete time hybrid systems - a case study», *Information Sciences*, vol. 177, n° 5, p. 1290–1308.

Boufaied, A. 2003, *Contribution à la surveillance distribuée des systèmes à événements discrets complexes*, thèse de doctorat, L'université Paul Sabatier de Toulouse.

Bousson, K., J. Steyer, L. Travé-Massuyès et B. Dahhou. 1998, «From a rule-base to a predictive qualitative model-based approach using automated model generation. application to the monitoring and diagnosis of biological process», *Engineering Applications of Artificial Intelligence*, vol. 11, p. 477–493.

Bouyer, P. et F. Chevalier. 2005, «Fault diagnosis using timed automata», dans *Foundations of Software Science and Computational Structures : 8th International Conference, FOSSACS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005*, Springer-Verlag, p. 4–8.

Bérard, B., A. Petit, V. Diekert et P. Gastin. 1998, «Characterization of the expressive power of silent transitions in timed automata», *Fundam. Inf.*, vol. 36, n° 2-3, p. 145–182, ISSN 0169-2968.

Cassandras, C. et S. Lafortune. 1999, *Introduction to Discrete Event Systems*, Kluwer Academic Publisher.

Cassez, F. et O. H. Roux. 2006, «Structural translation from time petri nets to timed automata», *Journal of Software and Systems*, vol. 79, p. 1456–1468.

Chen, Y. et G. Provan. 1997, «Modeling and diagnosis of timed discrete event systems-a factory automation example», dans *The American Control Conference, New Mexico*, p. 31–36.

Chow, E. et A. Wilsky. 1984, «Analytical redundancy and the designe of robust failure detection system», *IEEE transactions on Automatic and Control*, vol. 29, n° 7, p. 603–614.

Cocquempot, V., T. E. Mezyani et M. Staroswieckiy. 2004, «Fault detection and isolation for hybrid systems using structured parity residuals», dans *Asian Control Conference, ASCC'04*, vol. 2, New Mexico, p. 1204–1212.

Combacau, M. 1991, *Commande et surveillance des systèmes à événements discrets complexes : applications aux ateliers flexibles*, thèse de doctorat, L'université Paul Sabatier de Toulouse.

Combacau, M., P. Berrut, F. Charbonnaud et A. Khatab. 2000, «Refexions sur la terminologie : Surveillance - supervision», *Groupement pour la recherche en Produtique, Systèmes de Production Sûrs de Fonctionnement.*

Combacau, M., L. Kouiss et A. Toguyeni. 2002, *Fondements du pilotage des systèmes de production, Traité IC2 Productique*, Hermes Science.

Cordier, M., C. Largouët et C. Largou. 2001, «Using model-checking techniques for diagnosing discrete-event systems», dans *Proceedings of DX'01*, p. 39–46.

Cormen, T. H., C. E. Leiserson et R. L. Rivest. 1990, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts.

David, R. et H. Alla. 1989, *Du Grafcet aux réseaux de Petri*, Hermes Science Publications, Paris.

David, R. et H. Alla. 2004, *Discrete, Continuous, and Hybrid Petri Nets*, Springer, Berlin Heidelberg.

Debouk, R., S. Lafortune et D. Teneketzis. 2000, «Coordinated decentralized protocols for failure diagnosis of discrete event systems», *Discrete Event Dynamic Systems : Theory and Applications*, vol. 10, n° 1–2, p. 33–86.

Derbel, H., H. Alla, N. Ben Hadj-Alouane et M. Yeddes. 2009a, «Diagnosis of systems with rectangular hybrid automata models», *Soumis à AUTOMATICA.*

Derbel, H., H. Alla, N. Ben Hadj-Alouane et M. Yeddes. 2009b, «Online diagnosis of systems with rectangular hybrid automata models», dans *13th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'09*, Moscow, Russia, p. 123–129.

Derbel, H., N. Ben Hadj-Alouane et M. Yeddes. 2009c, «Diagnosis of a class of timed discrete event systems», *Soumis à Discrete Event Dynamic Systems*.

Derbel, H., N. Ben Hadj-Alouane, M. Yeddes et H. Alla. 2009d, «Limited-time lookahead diagnosability of rectangular hybrid automata», dans *3rd IFAC Conference on Analysis and Design of Hybrid Systems, ADHS'09*, Saragosse, Spain.

Derbel, H., M. Yeddes, N. Ben Hadj-Alouane et H. Alla. 2006, «Diagnosis of a class of timed discrete event systems», dans *8th International Workshop on Discrete Event Systems, WODES'06*, Michigan, USA, p. 256–261.

Deschamps, E. 2007, *Diagnostic de services pour la reconfiguration dynamique de systèmes à événements discrets complexes*, thèse de doctorat, Laboratoire d'Automatique de Grenoble.

Desel, J., W. Reisig et G. Rozenberg. 2004, «Lectures on concurrency and petri nets : advances in petri nets», *Lecture Notes on Concurrency and Petri Nets*, p. 87–124.

Domlan, E., D. Maquin et J. Ragot. 2004, «Diagnostic des systèmes à commutation, approche par la méthode de l'espace de parité», dans *Conférence Internationale Francophone d'Automatique, Tunisie*.

Dousson, C. 1994, *Suivi d'Evolutions et Reconnaissance de Chroniques*, thèse de doctorat, Laboratoire d'Analyse et d'Architecture des Systèmes, Toulouse.

Dousson, C. 2007, *Contribution à l'application de la reconnaissance des formes et la théorie des possibilités au diagnostic adaptatif et prédictif des systèmes dynamiques*, thèse de doctorat, Université de Reims Champagne-Ardenne.

Dubuisson, B. 1990, *Diagnostic et reconnaissance des formes*, Hermès.

Farreny, H. 1989, *Les systèmes experts - Principes et exemples*, Cépadues.

Feenstra, P. J., P. J. Mosterman, G. Biswas, P. C. Breedveld et F. O. E. Engineering. 2001, «Bond graph modeling procedures for fault detection and isolation of complex flow processes», dans *ICBGM'0133*, vol. 1, p. 77–82.

Feng, S. N., F. Zhao, G. Biswas et E. Hung. 2000, «Fault isolation in hybrid systems combining model based diagnosis and signal processing», dans *Proceedings of the 4th IFAC Symp. SAFEPROCESS*, p. 1074–1079.

Fourlas, K., K. Kyriakopoulos et N. Krikelis. 2002, «Diagnosability of hybrid systems», dans *Proceedings of MCCA'02*, p. 3994–3999.

Frehse, G. 2005, «Phaver : Algorithmic verification of hybrid systems past hytech», dans *Fifth International Workshop on Hybrid Systems : Computation and Control (HSCC)*, p. 258–273.

Gertler, J. et D. Singer. 1990, «A new structural framework for parity equation-based failure detection and isolation», *Automatica*, vol. 26, n° 2, p. 381–388.

Ghazel, M. 2005, *Surveillance des Systèmes à Evénements Discrets à l'Aide des Réseaux de Petri T-Temporels*, thèse de doctorat, l'Ecole Centrale de Lille et l'Université des Sciences et Technologies de Lille.

Ghazel, M., Toguéni et M. Bigang. 2005, «A monitoring approach for discrete events systems based on a timed perti net model», dans *16th IFAC World Congress*.

Gomaa, M. 1997, *Représentation et Supervision des Systèmes Hybrides par Réseaux de Petri*, thèse de doctorat, Institut national polytechnique de Grenoble, Grenoble.

Gomaa, M. et S. Gentil. 1996, «Hybrid industrial dynamical system supervision via hybrid continuous causal petri nets», dans *CESA'96 IEEESMC IMACS Symposium on Discrete Events and Manufacturing Systems, Lille, France*, Springer-Verlag, p. 380–384.

Grastien, A. 2005, *Diagnostic décentralisé et en-ligne de systèmes à événements discrets reconfigurables*, thèse de doctorat, Université Rennes 1.

Guéguen, H. et J. Zaytoon. 2004, «On the formal verification of hybrid systems», *Control Engineering Practice*, vol. 12, n° 10, p. 1253–1267.

Harel, D. et A. Pnueli. 1987, «Statecharts : A visual formalism for complex systems», .

Henzinger, T., P. Kopke, A. Puri et P. Varaiya. 1998, «The what's decidable about hybrid automata ?», *Journal of Computer and System Sciences*, vol. 57, p. 94–124.

Henzinger, T. A. 1996, «The theory of hybrid automata», *Hybrid Systems II, LNCS*, vol. 999, p. 278–292.

Henzinger, T. A., P. H. Ho et H. W. Toi. 1997, «Hytech : A model checker for hybrid systems», *International Journal on Software Tools for Technology Transfer*, vol. 1, n° 1-2, p. 110–122.

Henzinger, T. A. et R. Majumdar. 2000, «Symbolic model checking for rectangular hybrid systems», dans *TACAS 2000 : Tools and algorithms for the construction and analysis of systems, Lecture Notes in Computer Science, New-York*, Springer-Verlag, p. 142–156.

Isermann, R. 1984, «Process fault detection based on modeling and estimation methods – a survey», *Automatica*, vol. 20, n° 4, p. 387–404.

Jiang, S., Z. Huang, V. Chandra et R. Kumar. 2000, «A polynomial algorithm for testing diagnosability of discrete event systems», *IEEE Transactions on Automatic Control*, vol. 46, p. 1318–1321.

Jones, H. L. 1973, *Failure detection in linear systems.*, thèse de doctorat, Massachusetts Institute of Technology.

Karsai, G., S. Abdelwahed et G. Biswas. 2003, «Integrated diagnosis and control for hybrid dynamic systems», .

Kopke, P. W. 1996, *The Theory of Rectangular Hybrid Automata*, thèse de doctorat, Cornell University, NY, USA.

Koutsoukos, X., F. Zhao, H. Haussecker, J. Reich et P. Cheung. 2001, «Fault modeling for monitoring and diagnosis of sensor-rich hybrid systems», dans *Proceedings of the 40th IEEE Conference on Decision and Control*, p. 793–801.

Kurovszky, M. 2002, *Etude des sytèmes dynamiques hybrides par représentation d'état discrète et automate hybride*, thèse de doctorat, L'université Joseph Fourier, Grenoble.

Larsen, K. G., P. Pettersson et W. Yi. 1997, «Uppaal in a nutshell», *Journal of Software Tools for Technology Transfer*, vol. 1, n° 1-2, p. 134–152.

Lefebvre, D. 2000, «Contribution à la modélisation des systèmes dynamiques à événements discrets pour la commande et la surveillance», *Habilitation à Diriger des Recherches, Université de Franche Comté/ IUT Belfort - Montbéliard.*

Lin, F. et W. M. Wonham. 1988, «On observability of discrete-event systems», *Information sciences*, vol. 44, n° 3, p. 173–198.

Lin, F. et W. M. Wonham. 1994, «Diagnosability of discrete event systems and its applications», *Discrete Event Dynamic Systems*, vol. 4, n° 2, p. 197–212.

Lunze, J. 2000, «Diagnosis of quantised systems», dans *4th IFAC Symposium on Fault Detection. Supervision and Safety for Technical Processes, SAFEPROCESS'00*, p. 28–39.

Lunze, J. 2006, «Diagnosis of discretely controlled continuous systems», *Automatisierungstechnik*, vol. 54, n° 8, p. 385–395.

Merlin, P. M. 1974, *A study of the recoverability of computing systems*, thèse de doctorat, Department of Information and Computer Science, University of California.

Müller, O. et T. Stauner. 2000, «Modelling and verification using linear hybrid automata - a case study», *Mathematical and Computer Modelling of Dynamical Systems*, vol. 6, n° 1, p. 71–89.

Mokhtari, A. 2007, *Diagnostic des systèmes hybrides : développement d'une méthode associant la détection par classification et la simulation dynamique*, thèse de doctorat, Institut National des Sciences Appliquées de Toulouse.

Montmain, J. 1992, *nterprétation qualitative de simulation pour le diagnostic en ligne de procédé continus*, thèse de doctorat, Institut National Polytechnique de Grenoble.

Mosterman, J. 2001, «Diagnosis of physical systems with hybrid models using parameterised causality», dans *Hybrid Systems : Computation and Control, 4th International Workshop, HSCC'01*, Rome, Italia, p. 447–458.

Mosterman, P. J. 1997, *Hybrid Dynamic Systems : a Hybrid Bond Graph Modeling Paradigm and its Application in Diagnosis*, thèse de doctorat, Vanderbilt University.

Ondel, O. 2006, *Diagnostic par reconnaissance des formes : Application à un ensemble convertisseur-machine asynchrone*, thèse de doctorat, École Centrale de Lyon.

Pearson, J., C. Weise, W. Yi, G. Behrmann, G. Behrmann, K. G. Larsen et K. G. Larsen. 1998, «Efficient timed reachability analysis using clock difference diagrams», dans *Proceedings of the 12th Int. Conf. on Computer Aided Veri*, Springer-Verlag, p. 341–353.

Perrin, J., F. Binet, J. Dumery, C. Merlaud et J. Trichard. 2004, *Automatique et informatique industrielle : Bases théoriques, méthodologiques et techniques*, Nathan Technique.

Petri, C. A. 1962, *Kommunication mit automaten*, thèse de doctorat, Univeristy of Bonn.

Philippot, A. 2006, *Contribution au diagnostic décentralisé des systèmes à événements discrets : Application aux systèmes manufacturiers*, thèse de doctorat, L'Université de Reims Champagne Ardenne.

Ramadge, P. et W. Wonham. 1987, «Supervisory control of a class of discrete event systems», *SIAM J.Contr. Optim*, vol. 25, p. 57–96.

Rosnay, J. 1975, *Le macroscope : vers une vision globale*, Seuil.

Roux, O. et V. Rusu. 1996, «Uniformity for the decidability of hybrid automata», dans *Proceedings of the 8th Conference on COmputerAided Veri CAV'96, volume 1145 of LNCS*, p. 301–316.

Samantaray, A. K. et B. O. Bouamama. 2008, *Model-based Process Supervision*, chap. Diagnostic and Bicausal Bond Graphs for FDI.

Sampath, M., S. Lafortune et D. Teneketzis. 1998, «Active diagnosis of discrete-event systems», *IEEE Transactions on Automatic Control*, vol. 43, n° 7, p. 908–929.

Sampath, M., R. Sengupta, S. Lafortune et K. Sinnamohideen. 1996, «Failure diagnosis using discrete event models», *IEEE Transactions on Control Systems Technology*, vol. 4, n° 2, p. 105–124.

Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen et D. Teneketzis. 1995, «Diagnosability of discrete event systems», *IEEE Transactions on Automatic Control*, vol. 40, n° 9, p. 1555–1575.

Sava, A. T. et H. Alla. 2001, «Commande par supervision des systèmes à événement discrets temporisés», *Modélisation des systèmes réactifs, MSR'01*, p. 71–86.

Silveira, M. D. 2003, *Sur la distribution avec redondance partielle de modèles à événements discrets pour la supervision de procédés industriels*, thèse de doctorat, L'université Paul Sabatier de Toulouse.

Spathopoulos, M. 2000, «Supervisory control for rectangular hybrid automata», dans *39th IEEE Conference on Decision and Control*, p. 35–41.

Tagina, M., J. P. Cassar, G. Dauphin-Tangy et M. Staroswiecki. 1995, «Monitoring of systems modelled by bond graph», dans *International Conference on Bond Graph Modeling and Simulation, ICBGM'95*, Las Vegas, USA, p. 275Ű–280.

Thorsley, D. et D. Teneketzis. 2005, «Diagnosability of stochastic discrete-event systems», *IEEE Transactions On Automatic Control*, vol. 50, n° 4, p. 476–492.

Toguyeni, A. 1992, *Surveillance et diagnostic en ligne dans les ateliers flexibles de l'industrie manufacturière*, thèse de doctorat, Université de Lille.

Travé-Massuyès, L., P. Dague et F. Guerrin. 1997, *Le raisonnement qualitatif pour les sciences de l'ingénieur*, Hermès.

Tripakis, S. 1999, «Verifying progress in timed systems», dans *ARTS'99*, Springer-Verlag, p. 299–314.

Tripakis, S. 2002, «Fault diagnosis for timed automata», dans *Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT'02*, Springer-Verlag, London, UK, ISBN 3-540-44165-4, p. 205–224.

Valette, R., J. Cardoso et D. Dubois. 1989, «Monitoring manufacturing systems by means of petri nets with imprecise markings», dans *IEEE Conference Intelligent Control, NY*, p. 233–238.

Villemeur, A. 1988, *Sûreté de fonctionnement des systèmes industriels*, vol. 67, Edition EYROLLES, Collection DER-EDF.

Willsky, A. 1976, «A survey of design methods for failure detection in dynamic systems», *Automatica*, vol. 12, n° 6, p. 601–611.

Xue, F., D. Zheng et L. Yan. 2005, «Fault diagnosis of distributed discrete event systems using obdd», *IEEE Transactions On Automatic Control*, vol. 16, n° 3, p. 431–448.

Yovine, S. 1997, «Kronos : A verification tool for real-time systems», *International Journal on Software Tools for Technology Transfer*, vol. 1, p. 123–133.

Yovine, S. 1998, «Model checking timed automata», dans *European Educational Forum : School on Embedded Systems*, Springer-Verlag, p. 114–152.

Zad, S., R. Kwong et W. M. Wonham. 2005, «Fault diagnosis in discrete-events systems : Incorporating timing information», *IEEE Transactions On Automatic Control*, vol. 50, n° 7, p. 1010–1015.

Zad, S. H., R. H. Kwong et W. M. Wonham. 1998, «Fault diagnosis in discrete-event systems», dans *Proceedings of the IEEE Conference on Decision and Control (CDC'98)*, p. 3769–3774.

Zad, S. H., R. H. Kwong et W. M. Wonham. 1999, «Fault diagnosis in finite-state automata and timed discrete-event systems», dans *Proceedings of the 38th IEEE Conference on Decision and Control*.

Zad, S. H., R. H. Kwong et W. M. Wonham. 2003, «Fault diagnosis in discrete-event systems : Framework and model reduction», *IEEE Transactions On Automatic Control*, vol. 48, n° 7, p. 1199–1212.

Zemouri, M. R. 2003, *Contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques : Application à la e-maintenance.*, thèse de doctorat, l'Université de Franche-Comté.

Zwingelstein, G. 1995, *Diagnostic des défaillances. Traité des nouvelles Technologies, série Diagnostic et Maintenance*, Hérmes.

Résumé : Notre travail de recherche concerne l'étude du diagnostic à base de modèles pour les systèmes temporisés et pour une sous-classe de systèmes dynamiques hybrides. Nous avons d'abord développé une méthode de diagnostic basée sur la compilation hors-ligne d'un diagnostiqueur à partir du modèle automate temporisé du système à diagnostiquer. Une méthode systématique permettant la vérification de la diagnosticabilité du modèle utilisé est ensuite donnée. Nous avons ensuite proposé une méthode de diagnostic pour une sous-classe de systèmes dynamiques hybrides modélisés par des automates hybrides rectangulaires. Cette méthode repose sur l'utilisation d'une procédure de diagnostic en-ligne qui estime l'état courant du système ainsi que les occurrences des défauts non-observables. Enfin, nous avons proposé une méthode de vérification de la diagnosticabilité du langage temporisé accepté par un automate hybride rectangulaire vérifiant les hypothèses considérées dans notre travail.
Mots clefs : diagnostic, systèmes temporisés, systèmes dynamiques hybrides, automates temporisés, automates hybrides rectangulaires, analyse d'atteignabilité.

Abstract: In this thesis, we study the diagnosis of timed systems and a class of hybrid dynamic systems. A diagnosis approach for timed systems, using timed automata as system model, is given. This approach is based on the, off-line, construction of a timed automaton called diagnoser. The diagnoser allows to estimate unobservable failure occurrences, given a record of timed observable events, generated by the system. A systematic method for verifying the diagnosability of the studied model is also given. In the hybrid dynamic systems context, a diagnosis approach based on rectangular hybrid automata, is given. This approach consists on the execution of an on-line diagnosis procedure, which estimates, on-the-fly, the system state as well as failure occurrences. A diagnosability verification method of the studied hybrid model is provided.
Keywords : diagnosis, timed systems, hybrid dynamic systems, timed automata, rectangular hybrid automata, reachability analysis.

Laboratoire GIPSA-lab - ENSE^3 - BP 46, 38402 Saint-Martin d'Hères, FRANCE.