Détection et diagnostic basés cohérence pour les systèmes à événements discrets : vers la prise en compte des erreurs de modélisation

Carmen Guadalupe Lopez-Varela

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE TOULOUSE
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

THÈSE

en vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE
délivré par l'Institut National des Sciences Appliquées

Discipline : Systèmes Industriels

présentée et soutenue

par

Carmen Guadalupe LOPEZ-VARELA

le 17 décembre 2007

Détection et diagnostic basés cohérence pour les systèmes à événements discrets : vers la prise en compte des erreurs de modélisation

Directeurs de thèse

Audine SUBIAS

Michel COMBACAU

JURY

Mme. Louise Trave Massuyes	Présidente
M. Etienne Craye	Rapporteur
M. Eric Zamaï	Rapporteur
M. Pascal Berruet	Examinateur
M. Thierry Coudert	Invité

Remerciements

Le travail de recherche présenté dans ce document a été réalisé au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du Centre National de la Recherche Scientifique (CNRS). A ce titre, je tiens à remercier tout d'abord à M. Malik GHALLAB et M. Raja CHATILA (successivement) Directeur du LAAS, de m'avoir accueilli dans leur laboratoire.

Je remercie également M. Joseph AGUILAR-MARTIN et Mme. Louise TRAVE-MASSUYES (successivement) responsables du groupe de recherche de Diagnostic, Supervision et COnduite qualitatifs (DISCO), pour leur accueil dans ce groupe.

Je veux remercier très sincèrement mes directeurs de recherche Mme. Audine SUBIAS Maître de conférences à l'Institut National des Sciences Appliques (INSA) de Toulouse et M. Michel COMBACAU Professeur à la Université Paul Sabatier, pour la confiance, la sympathie et la patience qu'ils m'ont accordée tout au long de ces années de thèse. Je manifeste ma reconnaissance et gratitude envers eux pour m'avoir guidé, conseillé et encouragé au cours ce travail de recherche, ce qui m'a permis d'aboutir à la production de ce document.

Je tiens à exprimer mes remerciements aux membres du jury, qui ont accepté d'évaluer mon travail de thèse. Merci à Madame Louise TRAVE-MASSUYES Directeur de Recherche du CNRS, d'avoir accepté de présider le jury de cette thèse, à M. Etienne CRAYE Directeur à l'Ecole Centrale de Lille et M. Eric ZAMAÏ Maître de Conférences à l'Institut National Polytechnique de Grenoble INPG, d'avoir accepté d'être les rapporteurs de cette thèse. Leurs remarques et suggestions lors de la lecture de mon rapport m'ont permis d'apporter des améliorations à la qualité de ce dernier. Merci également à M. Pascal BERRUET Maître de conférences à l'Université de Bretagne Sud et M. Thierry COUDERT Maître de conférences à l'École Nationale d'Ingénieurs de Tarbes d'avoir accepté d'examiner mon mémoire et de faire partie de mon jury de thèse.

Je suis également reconnaissante de l'effort financier que mon pays, le Mexique, a fait pour me permettre de réaliser cette thèse. Cette aide financière a été gérée par la Dirección General de Educación Superior Tecnológica que je tiens à remercier vivement, ainsi que mon université l'Instituto Tecnológico de Hermosillo pour leur soutien pour toutes les démarches administratives.

Je souhaite remercier l'ensemble des services du LAAS et plus particulièrement Monsieur Christian BERTY pour son gentillesse et son efficacité lors de l'impression de ce document.

Je souhaiterais aussi remercier Yannick pour la détermination de 109 cas, très importantes dans ces travaux et à Xavier pour ses conseils en langage C et en Latex pour la

rédaction de ce document. Je veux aussi manifester aux thésards et permanents de mon groupe de recherche, le plaisir que j'ai eu de partager avec eux de bon moments au cours des ces années.

Je voudrais remercier mes amis desquels j'ai reçu beaucoup de soutien des mon arrivé à Toulouse : Guillermo, Rebeca, Alberto, Fidel, Maria Elena, Luisita, et plus spécialement Lucila, Ruth et Niriaska, Iraima, Alejandra pour leur aide et amitié inconditionnelle.

Finalement j'adresse un grand merci à toute ma famille, ma mère, mon père et mes frères qui m'ont soutenue jusqu'à la fin de cette aventure.

Très spécialement, j'exprime la plus grande gratitude à mon mari et mon fils, qui ont été la motivation plus importante et qui ont toujours été présents lorsque j'en ai eu besoin et qui ont suivi avec moi tous les étapes nécessaires pour pouvoir accomplir cette thèse.

Table des matières

Introduction

L'automatisation des installations à permis d'augmenter la productivité des systèmes pour répondre aux exigences du marché, mais en même temps elle a augmenté la complexité des moyens de production. Cette sophistication ne suffit pas pour garantir les niveaux de production et de qualité requis. Il est également nécessaire d'assurer la disponibilité et la performance des moyens de production en détectant la présence des défaillances pouvant affecter le système. La détection des défaillances permet en effet d'envisager des actions correctives et de remettre le plus tôt possible le système dans un état fonctionnel requis.

Différentes approches de surveillance ont été développées par les communautés scientifiques avec le but de détecter les défaillances et d'en corriger les conséquences au niveau du système.

Dans ces approches, le rôle de la détection est d'identifier toute manifestation de défaillance indiquant un éloignement du comportement du procédé par rapport à son comportement nominal. L'état du procédé se rapproche alors d'états non souhaités indiquant des dégradations de la performance ou de la sûreté du système. Après la détection de ces manifestations observables appelées symptômes, la fonction diagnostic est chargée généralement de localiser les composants défaillants et d'identifier les causes de ces manifestations. Autrement dit, le diagnostic se charge de relier les causes aux effets. Cette fonction est aussi censée donner toutes les explications nécessaires sur l'apparition de la défaillance, explications qui sont ensuite utilisées pour la mise en place d'actions correctives. Dans certaines approches la fonction détection est considérée intégrée dans la fonction diagnostic et dans d'autres approches, comme la nôtre, la fonction détection est une fonction séparée de la fonction diagnostic.

De façon générale, il est possible de distinguer les approches utilisant un modèle du système dans lequel le modèle capture la connaissance sur le système et les approches à base de données, qui résultent de la collecte d'informations sur le système surveillé.

Dans le cas des systèmes à événements discrets qui font l'objet de ce travail, les approches de diagnostic à base de modèles ont aussi évolué au fur et à mesure qu'ont évolué les systèmes à surveiller. L'approche la plus connue dans ce domaine est l'approche diagnostiqueur qui se base sur un modèle de fautes. Cette approche à l'origine centralisée a évolué de nos jours vers des versions décentralisées ou distribuées pour mieux s'adapter aux systèmes à surveiller. Dans ce domaine, un diagnostic est défini comme l'ensemble des trajectoires (états et événements) qui expliquent les observations issues du système.

Néanmoins, dès les origines du diagnostic à base de modèles s'est posé le problème de l'obtention des modèles. Une grande majorité des approches développées s'appuie sur l'existence d'un modèle de fautes ou de bon comportement, modèle supposé correct par hypothèse. Dans le cas d'un modèle de fautes, celui-ci est supposé en plus être exhaustif.

Il est toutefois réaliste d'envisager que les modèles utilisés puissent être incomplets et/ou incorrects.

Plusieurs facteurs peuvent empêcher d'obtenir un modèle adéquat pour le diagnostic, que ce soit un modèle représentant les défaillances ou un modèle capturant le comportement normal du système. La complexité croissante des moyens de production est un de ces facteurs : la non maîtrise ou l'incompréhension du fonctionnement du système peut être à la base d'erreurs de modélisation.

Ces erreurs de modélisation vont bien entendu compliquer les étapes de détection de défaillances et de diagnostic. Si ces erreurs se situent au niveau d'un modèle de bon comportement, elles peuvent provoquer une détection intempestive et donc la détection d'un faux symptôme ou bien au contraire masquer un comportement défaillant.

Cette prise en compte des erreurs de modélisation dans les modèles utilisés par la détection et le diagnostic est au cœur du travail que nous présentons dans ce document et en fait une de ses principales originalités.

Nous proposons une approche à base de modèles de bon fonctionnement du système. Ce choix vise deux objectifs : d'une part de ne pas s'appuyer nécessairement sur la connaissance des modes de défaillance du procédé et, d'autre part, d'utiliser les modèles issus des phases de conception pour aller vers une conception conjointe du système et de sa surveillance. Nous présentons dans le premier chapitre de cette thèse la théorie de base pour la surveillance-supervision de systèmes à événements discrets. Dans le deuxième chapitre nous présentons l'état de l'art sur le diagnostic de défaillances dans les systèmes à événements discrets, en particulier l'approche diagnostiqueur qui se base sur un modèle décrivant les défaillances du système et, nous situons notre contribution par rapport à ces travaux.

Dans le chapitre trois, nous donnons une vision générale de notre approche avec une brève présentation des différentes étapes mises en place. La première étape est ensuite détaillée dans le chapitre 4, il s'agit de la détection. Celle-ci repose sur deux modèles du système représentant uniquement le comportement en fonctionnement normal. Nous modélisons en ligne le comportement observé pour le confronter avec les comportements prévus par les modèles.

Le chapitre 5 est consacré à la deuxième étape de notre approche qui est l'identification des configurations générées par les erreurs dans les modèles du système. Cette étape doit permettre de distinguer les symptômes dus à des erreurs de modélisation de ceux dus à de réelles défaillances au niveau du procédé.

Enfin, la dernière partie de notre approche est la fonction diagnostic et nous la présentons dans le chapitre 6 de ce document. Nous présentons le diagnostic comme la modification des modèles de manière à éliminer les incohérences à l'origine de la détection d'un symptôme. Le chapitre 7 qui est le dernier chapitre de ce document est dédié à l'application de chacune des étapes de notre approche de détection et diagnostic basée cohérence sur un exemple issu du domaine des systèmes de production.

Chapitre 1

Commande, Supervision et Surveillance des Systèmes à Evénements Discrets

Introduction

Dans ce chapitre nous présentons le contexte général de notre travail de recherche. Nous nous intéressons à la surveillance des systèmes à événements discrets pour lesquels l'évolution entre états du système est provoquée par des événements dont l'occurrence n'est pas forcément périodique. De nombreux travaux se focalisent plus particulièrement sur le diagnostic des défaillances qui affectent le comportement normal de ces systèmes. Les systèmes informatiques, les systèmes de production, les systèmes de transport ou bien encore les réseaux de télécommunications sont quelques exemples de systèmes à événements discrets auxquels ces travaux s'appliquent. Bien que nos travaux aient une portée plus générale, nous concentrons notre attention sur un seul domaine d'application : les systèmes flexibles de production manufacturière.

Nous présentons dans ce chapitre tous les concepts de base utilisés dans de notre travail de recherche comme la commande, la supervision et la surveillance de systèmes à événements discrets et les relations entre ces trois concepts. Nous détaillons ensuite les fonctions détection et diagnostic de défaillances sur lesquelles porte notre travail de recherche. Finalement nous présentons les systèmes de production manufacturière.

1.1 La commande

La fonction commande se charge de contrôler le système en appliquant une séquence d'activités de commande (ou ordres) à exécuter pour assurer la réalisation d'un produit ou service qui contribue à atteindre un objectif en particulier.

Ainsi, sur la base des travaux réalisés par [Combacau, 1991], un système de commande est défini comme un système incluant des fonctions agissant directement sur le système commandé appelé procédé. Le rôle de ce système est d'activer l'exécution d'un ensemble d'opérations au niveau du procédé. Ceci se fait au travers des ordres envoyés aux actionneurs du procédé qui permettent d'amener le procédé vers un état requis par

l'utilisateur. Ces ordres répondent à certaines conditions de fonctionnement du procédé (normal, anormal,...). Quatre modes de fonctionnement peuvent être assurés par le système de commande :

- **le fonctionnement en absence de défaillances** : la loi de commande correspond à une séquence d'opérations associée à un comportement particulier permettant de répondre aux attentes d'un client. Il peut s'agir par exemple d'opérations constituant la gamme de fabrication d'un produit ;
- **la reprise ou gestion de modes** : il s'agit d'une séquence d'actions curatives acheminant le procédé vers un fonctionnement normal ;
- **le traitement d'urgence** : il correspond à l'application des actions prioritaires sur le procédé, lesquelles sont souvent prédéfinies dans l'objectif d'assurer la sécurité de l'installation et du personnel pour éviter des situations catastrophiques ;
- **la maintenance préventive** : qui prend en compte des opérations de test, de réglage, de nettoyage ..., permettant de garantir que le système pourra continuer à assurer sa mission.

Dans le domaine des systèmes à événements discrets, la commande fait évoluer le procédé d'un état vers un autre en passant par un certain nombre d'états intermédiaires observables [Zamaï, 1997]. Trois types d'états ont été déterminés dans [Combacau, 1991] :

- **les états interdits** à partir desquels certaines lois du fonctionnement correct des composants du procédé risquent d'être violées (résistance de matériaux, capacité limitée, etc) ;
- **les états utilisables** qui sont des états observables non interdits et qui correspondent au fonctionnement normal du procédé ;
- **les états autorisés**, il s'agit d'états associés à la satisfaction d'une commande particulière.

Dans ce cadre, l'application d'une commande consiste alors à amener le procédé vers les états autorisés qui satisfont la requête imposée par l'utilisateur du système.

La complexité des systèmes à commander a naturellement entraîné celle des systèmes de commande. Ceux-ci sont devenus de plus en plus sophistiqués et aujourd'hui le but est de développer des systèmes allant au delà de la commande, des systèmes capables de détecter des changements indésirables dans les procédés et d'isoler leur impact au niveau du procédé [Vu Trieu *et al.*, 2007].

En effet, les systèmes commandés sont souvent sujets à des événements imprévisibles tels qu'une défaillance d'un composant pouvant entraîner la dégradation de leurs performances générales [Berruet *et al.*, 2002]. Une défaillance est : *un événement caractérisant une situation dans laquelle une opération n'est pas exécutée par une ressource car son état n'est compatible avec aucune spécification nominale* [Combacau *et al.*, 2000].

Ce besoin de réaliser les objectifs tout en prenant en compte ces événements imprévisibles et en corrigeant leurs conséquences est à l'origine des concepts de surveillance et de supervision que nous présentons maintenant.

1.2 La surveillance

Le rôle de la surveillance est de veiller sur les évolutions du comportement du procédé et de collecter des informations pertinentes pour la prise de décisions dans le cas d'une défaillance. La surveillance rassemble des données en provenance du procédé lui permettant de déterminer l'état actuel du système surveillé et de faire les inférences nécessaires pour la production d'informations additionnelles comme le diagnostic de défaillances et les historiques du comportement observé. Ainsi, la surveillance assure la détection d'une anomalie du comportement mais également la collecte et la génération de toutes les informations indispensables pour remettre à nouveau le procédé en fonctionnement normal. L'ensemble des fonctions ou tâches lui permettant d'assurer son rôle sont les suivantes [Combacau, 1991] [Zamaï, 1997] [Combacau *et al.*, 2000].

- **L'acquisition de données**, qui correspond à la collecte des informations issues des capteurs du procédé et de son système de commande.
- **La perception**, il s'agit de l'extraction d'indicateurs à partir de données techniques pour aider à l'identification des symptômes de défaillances i.e des manifestations des défaillances.
- **La détection**, qui détermine la normalité ou l'anormalité du fonctionnement du système. Deux classes de situations anormales peuvent être considérées [Combacau *et al.*, 2000].

  1. La première regroupe des situations dans lesquelles les conditions opérationnelles du procédé ne sont plus respectées. Il peut s'agir par exemple de la violation d'une contrainte d'exclusion mutuelle entre deux bras manipulateurs.

  2. La deuxième catégorie englobe des situations où les lois de commande sont violées comme le dépassement des dates de fin de fabrication d'un produit.

- **Le diagnostic**, qui cherche les liens de causalité entre les symptômes observés, la défaillance et son origine. Cette fonction peut être divisée en trois sous-fonctions.

  - La localisation est chargée de déterminer le sous-système à l'origine d'un symptôme observé.
  - L'identification détermine les causes d'une défaillance.
  - L'explication fournit les conclusions du diagnostic.

- **Le pronostic**, permet entre autres de connaître les conséquences d'une défaillance diagnostiquée.
- **Le suivi**, maintient un enregistrement de l'historique des événements observés et des traitements exécutés par le système de commande/supervision.
- **L'urgence**, qui considère les situations critiques où la défaillance détectée traduit la transgression d'une contrainte structurelle ou la mise en danger de l'opérateur.

La fonction urgence dicte les procédures prédéfinies et prioritaires à appliquer pour éviter des évolutions dangereuses.

Avec cet ensemble de fonctions, un système de surveillance possède théoriquement la capacité de traiter les défaillances dont les manifestations sont observables directement ou indirectement. L'efficacité du traitement des défaillances dépend fortement de la façon dont les fonctions composant le système de surveillance sont organisées [Zamaï, 1997].

Cette vision de la surveillance nous amène à considérer la détection comme une fonction séparée de la fonction de diagnostic. Autrement dit, la détection n'est pas une étape du diagnostic. En effet, le diagnostic est vu comme un raisonnement permettant de trouver les liens entre les symptômes détectés préalablement par la fonction détection et les causes de ces symptômes.

En s'appuyant sur les informations fournies par la surveillance, la supervision va pouvoir prendre les décisions qui s'imposent en situation de défaillance.

1.3 La supervision

La supervision se charge de contrôler et de surveiller l'exécution d'une opération ou d'un travail effectué par d'autres sans rentrer dans les détails de cette exécution. La supervision agit sur le fonctionnement normal et anormal du système en agissant directement sur les modèles de la commande [Combacau *et al.*, 2000] :

- **en fonctionnement normal** la supervision agit en temps réel et prend les dernières décisions correspondant aux degrés de liberté exigés par la flexibilité décisionnelle. La flexibilité décisionnelle traduit en effet la possibilité de pouvoir répondre aux consignes du client selon la capacité et la disponibilité des ressources (composants) du système. La supervision réalise l'ordonnancement en temps réel [Henry, 2005] de manière à prendre les dernières décisions en fonction de la capacité des ressources ;
- **en présence de défaillances** la supervision prend toutes les décisions nécessaires pour faire face aux défaillances et assurer le retour du procédé vers un fonctionnement nominal permettant d'assurer la production. Il peut s'agir de choisir une solution curative, d'effectuer des réordonnancements locaux, de prendre en compte la stratégie de surveillance de l'entreprise [Hernandez De Leon, 2006], de déclencher des procédures d'urgence, etc.

Ainsi, le rôle de la supervision est décisionnel en même temps qu'opérationnel. La supervision impose des choix de fonctionnement, élabore des solutions de reconfiguration en réponse aux données structurées fournies par la surveillance et se sert du système de commande pour appliquer ces solutions [Zamaï, 1997] [Toguyeni, 1992].

Comme nous le présentons dans la section suivante, l'intégration de ces trois concepts (commande, supervision et surveillance) au sein d'une architecture gérant le fonctionnement d'un procédé a été adaptée à l'évolution des systèmes à gérer.

1.4 Intégration de la surveillance-supervision et de la commande

Les fonctions de surveillance-supervision peuvent être considérées comme intégrées ou séparées du système de commande et des approches mixtes sont aussi envisageables [Combacau, 1991] [Chaillet-Subias, 1995] [Zamaï, 1997].

- **Dans une approche intégrée**, les modèles de la commande incluent les états de défaillances ainsi que les traitements correctifs. Les comportements normaux et anormaux sont alors traités de la même façon. Dans ce cas, toutes les évolutions possibles doivent être prévues à l'avance et à chaque évolution normale ou anormale un traitement spécifique est associé. L'efficacité d'une telle approche dépend fortement de l'exhaustivité du recensement des évolutions possibles du procédé lors de son exploitation. Ramadge et Wonham ont proposé dans [Ramadge et Wonham, 1989] une méthode générique de synthèse de commande supervisée qui est un exemple d'une approche intégrée dans le domaine des systèmes manufacturiers. L'approche consiste à interdire les événements pouvant amener le procédé vers des évolutions dangereuses ou indésirables.
- **Dans l'approche séparée**, la commande se charge uniquement du fonctionnement normal du procédé alors que la supervision-surveillance prend en charge tous les fonctionnements en dehors du fonctionnement normal. Le problème inhérent à une telle approche est celui des conflits entre les actions à réaliser sur le procédé dans un objectif de commande et celles dictées par la surveillance-supervision. En effet, les actions engendrées par la surveillance-supervision ne seront pas normales pour la commande. Une approche séparée est présentée dans les travaux de [Holloway et Krogh, 1990] et appliquée aux systèmes manufacturiers.
- Une solution à ce problème consiste à adopter **une approche mixte** où par exemple la détection et la reprise sont intégrées à la commande, et les autres fonctions sont séparées. Dans ce cas, la fonction détection est implicite dans la commande mais il n'y a pas besoin de prévoir les évolutions anormales car l'approche considère anormales toutes les évolutions qui ne peuvent être traitées par la commande.

Nos travaux sont orientés vers les fonctions de surveillance et portent spécifiquement sur les fonctions détection et diagnostic que nous allons présenter plus en détail dans la partie suivante. Notre intérêt se portant sur les systèmes à événements discrets nous n'avons pas ici mentionné les approches provenant de la théorie du contrôle i.e. les approches de Fault Detection and Isolation, qui s'appuient généralement sur des modèles de fautes [Betta et Pietrosant, 2000] [Gertler, 1998] [Isermann, 1995] [Graton, 2005].

1.5 La fonction détection

Une défaillance est reconnue par ses manifestations. La fonction détection a pour but d'identifier ces manifestations observables appelées symptômes [Niel et Craye, 2002]. La détection d'un symptôme de défaillance est une étape indispensable pour envisager

le diagnostic de cette défaillance [Milne, 1987]. Pour cette raison, la fonction détection peut être considérée comme la fonction la plus importante de la surveillance : tout traitement de défaillance repose sur la détection de cette défaillance. Ainsi, la détection de la manifestation d'une défaillance précède la détermination des causes de cette défaillance ; quand il n'y a pas de manifestations observables, la détection n'est pas possible et le diagnostic n'a pas raison d'être.

Pour assurer cette détection, le système de surveillance s'appuie sur un ensemble de capteurs fournissant des indicateurs distribués stratégiquement dans le système surveillé afin d'obtenir des informations pertinentes sur celui-ci mais également sur son environnement [Betta et Pietrosant, 2000].

L'efficacité de la détection dépend très fortement de la capacité de distinction entre un comportement normal et un comportement anormal. Pour cette distinction la fonction doit disposer de paramètres ou de critères fiables permettant l'évaluation des comportements observés. Si les paramètres sont mal choisis, il est possible de détecter de faux symptômes et d'associer à un comportement normal un symptôme de défaillance ou bien de masquer certaines défaillances et d'accepter comme normal un comportement qui en réalité ne l'est pas.

Dans le contexte des systèmes à événements discrets, la détection des symptômes de défaillances est en général réalisée en analysant les événements observables issus du fonctionnement du procédé [Ghazel, 2005]. Ces défaillances peuvent être de deux types [Boufaied, 2004] :

1. **défaillances liées à la commande**, il s'agit de commandes que le procédé ne peut pas reconnaître et qui ont pour conséquence de provoquer des évolutions inattendues ;

2. **défaillances physiques des composants du procédé**, il s'agit de dysfonctionnements des composants (i.e. vieillissement, problèmes de réglage, calibrage, etc.) qui modifient le fonctionnement attendu du procédé.

Le principe de base de la détection est de confronter les évolutions observées du système avec celles d'un modèle qui évolue de façon synchrone avec le procédé [Valette et Künzle, 1994].

Un modèle mis en place dans un but de détection peut prendre en compte différents aspects (aspects logiques, séquentiels et temporels [Niel et Craye, 2002]) selon le type de défaillance visé. Les mécanismes de détection implémentés dépendent évidemment de ce choix. Deux types de modèles peuvent être utilisés pour la détection de comportements anormaux du procédé :

– modèle de fonctionnement normal : dans ce cas la détection est réalisée par la comparaison des évolutions observées du procédé et du modèle. Dans ce contexte, une incohérence entre l'état du modèle et l'évolution du système traduit une anomalie. Il est considéré en général qu'une anomalie peut être due à un dysfonctionnement du système ou à une erreur de modélisation. Ce dernier aspect n'est cependant pas traité dans la plupart des travaux sur la détection comme nous le verrons par la suite ;

– modèle de fonctionnement anormal : la détection est réalisée par la reconnaissance des défaillances inscrites dans le modèle à partir des observations issues du procédé.

La figure 1.1 montre une classification des techniques de détection en fonction du type de modèle utilisé [Combacau, 2002].

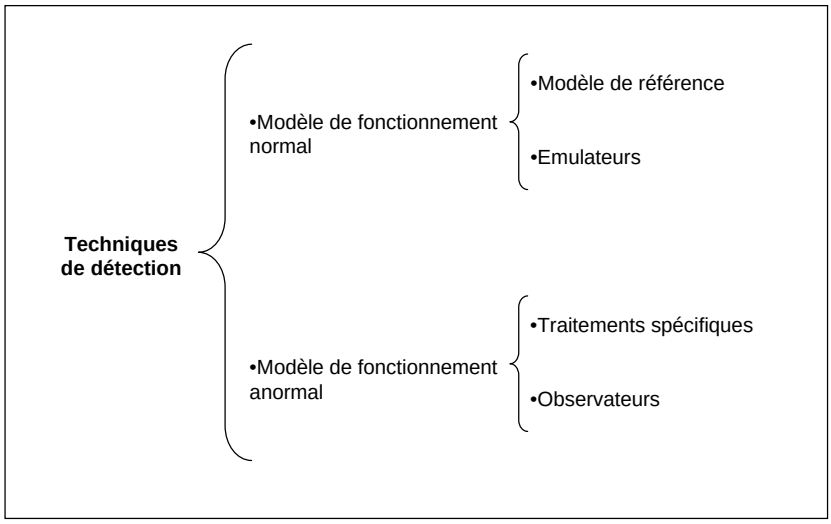


Fig. 1.1 – Classification des techniques de détection

La première catégorie de technique qui concerne les techniques utilisant un modèle du comportement regroupe deux approches : les approches à base d'émulateurs et celles à base de modèle de référence. Un émulateur reçoit les consignes et les signaux en provenance des capteurs et détermine l'état prévu dans lequel doit se trouver le procédé [Holloway et Krogh, 1990]. Ainsi, la détection est effectuée à l'apparition d'une déviation entre ce qui est prévu et ce qui est observé. La technique à modèle de référence utilise un modèle de commande comme une référence de fonctionnement normal [Combacau, 1991]. Les événements qui ne sont pas prévus dans de ce modèle sont interprétés comme des manifestations de défaillances.

Dans la deuxième catégorie nous trouvons des techniques utilisant un modèle de fonctionnement anormal. Les approches à base de traitements spécifiques et celles à base d'observateurs font partie de cette catégorie. Les approches dites à base de traitements spécifiques consistent à équiper le procédé de capteurs spécifiques permettant directement l'identification d'une défaillance. Alors, à l'observation d'un signal émis par un de ces capteurs, une défaillance est identifiée. Dans une approche à base d'observateurs des évolutions caractéristiques de défaillance sont identifiées à partir des informations issues du procédé et du système de commande. Un observateur contient alors le modèle de ces évolutions qui constituent les séquences de défaillances à reconnaître. Bien entendu l'observateur n'est capable de reconnaître que les séquences décrites dans le modèle.

Le choix d'une technique en particulier dépend fortement du système à surveiller. Quoi qu'il en soit, toute détection d'un symptôme de défaillance déclenche un processus de diagnostic afin de comprendre le pourquoi de ce symptôme.

1.6 La fonction diagnostic

Les défaillances dans les systèmes ont des impacts négatifs sur la disponibilité des moyens de production et même parfois sur la sûreté des installations et du personnel. Ainsi, une défaillance éloigne le procédé du comportement requis pour la réalisation des objectifs pour lesquels il a été conçu.

Il est donc primordial de diagnostiquer une défaillance pour déterminer ensuite les actions à entreprendre et supprimer efficacement ces effets négatifs.

Afin de ramener le procédé vers un comportement attendu, les actions correctives (réparation, changement de composants, etc.) doivent être menées sur les composants à l'origine de la défaillance. Le diagnostic doit donc localiser ces composants défaillants. En ce sens, l'étape du diagnostic est importante pour la maintenance corrective qui se charge de la réparation des composants qui ont été objet de défaillances [Zwingelstein, 1995] [Toscano, 2005]. Cette localisation doit idéalement être suivie de la détermination de la cause de la défaillance.

La complexité de la tâche de diagnostic a motivé la recherche de son automatisation. Les méthodes proposées dans la littérature pour réaliser un diagnostic incluant la fonction détection sont nombreuses. Ces méthodes varient selon le type de connaissance du système utilisé, selon la façon de structurer cette connaissance et de l'utiliser lors de la génération d'un diagnostic. Il est donc possible de classer les méthodes de diagnostic selon l'un ou l'autre de ces trois aspects [Zwingelstein, 1995] [Vu Trieu *et al.*, 2007] [Dubuisson, 2001]. La classification que nous présentons figure 1.2 est reprise de [Papadopoulos et McDermid, 2001] et est basée sur le type de connaissance utilisé pour le diagnostic de défaillances. Cette classification qui ne prétend pas être exhaustive va tout de même nous permettre de situer nos travaux.

Trois grandes catégories de méthodes sont identifiées dans cette classification : les approches à base de règles, les approches à base de modèles et les approches à base de données.

1.6.1 Les approches à base de règles

Ce type d'approche a pour origine le diagnostic médical et le système expert MYCIN [Buchanan *et al.*, 1984]. Un système expert est un logiciel qui reproduit la connaissance et le raisonnement d'un expert dans l'accomplissement d'une tâche. La connaissance utilisée dans ce système est formalisée par un ensemble de règles qui sont enchaînées pour simuler le raisonnement de l'expert du système. Un système expert peut se décomposer en deux parties :

- **la base de connaissance**, est composée d'une base de faits et d'un ensemble de règles. La base de faits contient les informations concernant les cas à traiter. Les règles appelées règles de production sont de la forme : Si X alors Y, où X est une conjonction de conditions et Y est une conclusion. Typiquement les règles décrivent les relations fonctionnelles entres les différents composants du système et les relations causales entre les défaillances et leurs effets. Ces règles sont formulées à partir des associations empiriques réalisées par les experts et portent sur les

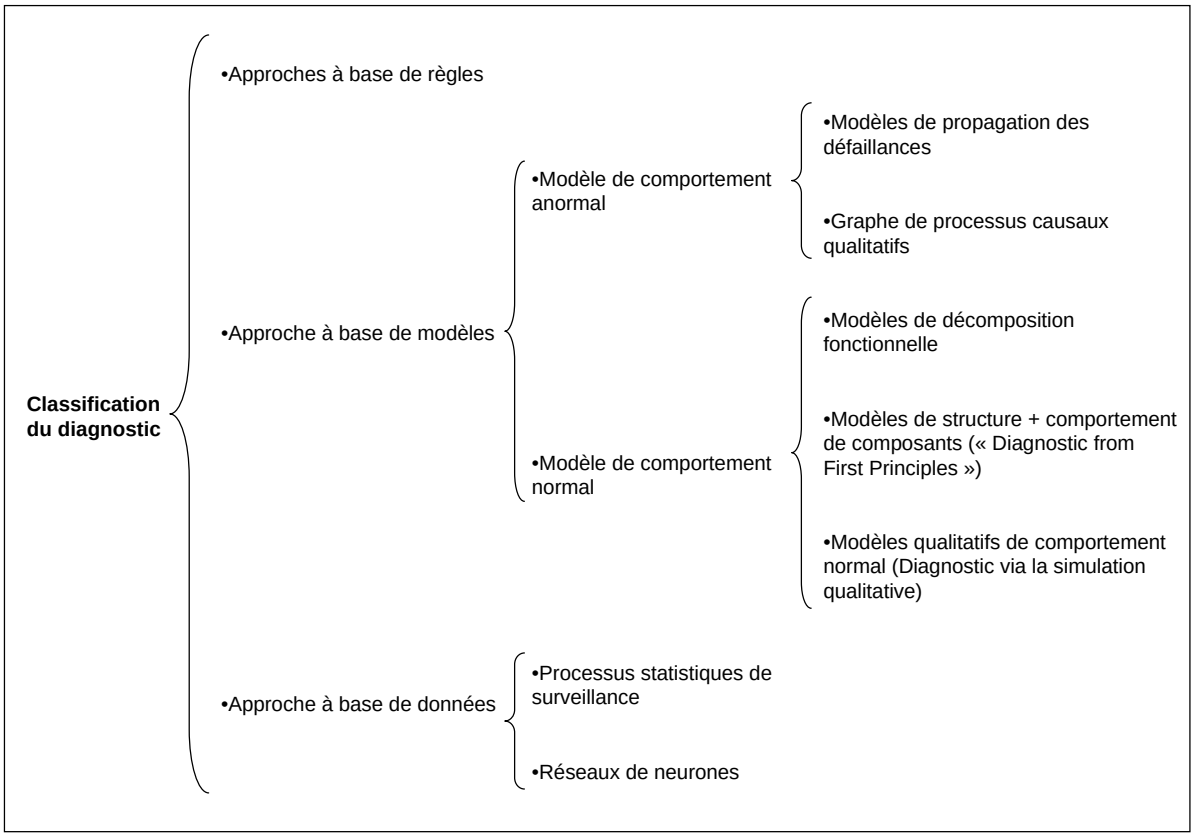


FIG. 1.2 – Classification des techniques de diagnostic

dysfonctionnements du système. Ces règles permettent de raisonner à partir des faits du système.

– **le moteur d'inférence**, qui permet d'atteindre un diagnostic en raisonnant à partir des données contenues dans la base de connaissances. Le raisonnement peut utiliser deux types de chaînage. Le chaînage avant qui permet à partir d'une déviation d'activer certaines règles. Quand une règle est activée (on dit aussi *tirée*) ses conclusions ou conséquences sont considérées comme de nouveaux faits qui activent d'autres règles considérant ce fait comme un antécédent. Cette procédure se poursuit jusqu'à ce qu'il n'y ait plus de règles à activer. Le chaînage arrière est un mécanisme d'induction, qui permet de raisonner à partir d'un fait en particulier et de rechercher toutes les règles qui ont comme conclusion ce fait. Les antécédents de ces règles deviennent ensuite les faits à partir desquels le raisonnement se poursuit.

L'avantage d'un tel système se situe dans l'indépendance qui existe entre la base de connaissances et le moteur d'inférence car la représentation de la connaissance n'est pas liée à la façon dont cette connaissance est utilisée par le moteur d'inférence ; ainsi la modification de la base de connaissances ne change pas le raisonnement du moteur d'inférence. Dans [KIM *et al.*, 2005] un système expert à base de règles est implémenté dans une architecture d'agents distribués qui collaborent pour la réalisation du diagnostic.

Dans le domaine automobile ce type d'approche a été aussi largement utilisé dans le but d'effectuer un diagnostic plus systématique et plus intelligent, car en plus de localiser les composants défaillants et leurs causes, le système propose les actions pertinentes pour

corriger les défaillances [Gelgele et Wang, 1998].

La principale limitation de ce type d'approche concerne son application pour le diagnostic des systèmes complexes. En effet le formalisme des règles permet difficilement de décrire toutes les propriétés d'un système. Toute l'expérience ne peut donc être représentée sous la forme d'une règle [Papadopoulos et McDermid, 2001]. De plus, l'approche présente une très forte dépendance au système : pour de nouveaux systèmes, de nouveaux ensembles de règles doivent être élaborés et un temps important doit donc être consacré à l'acquisition de l'expérience des dysfonctionnements potentiels du système [Davis et Hamscher, 1988].

1.6.2 Les approches à base de modèles

Les techniques du Diagnostic à Base de Modèles (DBM) issues de la communauté de l'intelligence artificielle sont fondées sur une théorie logique [Reiter, 1987] [Poole, 1989] [Dubuisson, 2001]. Dans ces approches la détection est considérée comme une tâche du diagnostic. Les premiers travaux s'appuyaient sur des associations de connaissances empiriques, comme ce qui est fait dans les systèmes experts. Pour pallier les limitations de la non-exhaustivité de l'expertise humaine, d'autres techniques de diagnostic à base de modèles plus élaborés ont été développées [Charbonnaud, 1992]. Ces approches utilisent des modèles basés sur la connaissance du système à diagnostiquer. De façon générale, deux types d'approches de DBM peuvent être distinguées [Dubuisson, 2001] : celles s'appuyant sur un modèle de comportement anormal (fautes) et celles qui reposent sur des modèles du comportement normal (bon comportement) du système.

- **Modèle de comportement anormal**, ce modèle est une description du comportement du système en présence de certaines défaillances affectant son fonctionnement. Certains de ces modèles prennent en compte également les propagations de défaillances. Les diagrammes Cause-Conséquences [Y.Papadopoulos, 2002] par exemple, représentent de façon formelle et logique, les relations causales entre les états des différents composants d'un système en situation de défaillance. Les Graphes de Processus Causaux Qualitatifs sont une représentation graphique des interactions entre les variables du procédé en situation de défaillance. Les "digraph" [Tateno *et al.*, 2006] [Kelly et Bartlett, 2006], "logic flowgraph" [Muthukumar *et al.*, 1991] sont des exemples de ce type de graphes, utilisés pour le diagnostic.
- **Modèle de comportement normal**, le modèle décrit uniquement comment se comporte le système quand il fonctionne correctement. De nombreux travaux dans ce domaine sont connus sous l'appellation de "diagnostic from first principles" [Reiter, 1987] [De Kleer et Williams, 1987] [Davis, 1984] [Genesereth, 1984] et s'appuient sur un modèle de la structure du système et du comportement de ses composants, pour effectuer des prédictions sur les états du système. Dans ce cas, le diagnostic consiste à déterminer les interactions entre les observations et les prédictions. Toute divergence donne lieu à la détection d'un symptôme de défaillance. L'ensemble de composants supposés défaillants qui peuvent

causer ces divergences constitue le diagnostic proprement dit. D'autres types de modèles existent. Certains comme les arbres d'objectif/arbre succès (GTST), se basent sur une décomposition fonctionnelle de l'objectif global du système en un ensemble de sous-objectifs, lesquels sont aussi décomposés jusqu'au niveau des composants physiques nécessaires pour satisfaire les sous-objectifs. Les combinaisons logiques des sous-objectifs garantissent la réalisation de l'objectif global. Dans des approches par simulation qualitative, comme celles basées sur l'outil QSIM, le modèle du système est sous forme d'un ensemble d'équations différentielles. Dans l'approche de diagnostic [Dvorak et Kuipers, 1989], après la détection d'une observation anormale, un ensemble d'hypothèses de défaillance est formulé, l'outil QSIM est utilisé pour prédire les comportements du système et ensuite comparer les prédictions avec les hypothèses de défaillances. Les hypothèses de défaillances qui correspondent aux prédictions sont retenues, les autres sont éliminées.

L'efficacité d'une approche basée sur un modèle de comportement anormal dépend de la capacité à anticiper les défaillances pouvant affecter le système surveillé, puisque seules les défaillances inscrites dans le modèle sont détectables. Le problème de ce type d'approche se situe donc au niveau de l'exhaustivité dans l'anticipation des défaillances, le nombre de défaillances d'un système étant potentiellement très important. La difficulté des approches basées sur un modèle de comportement normal est de décrire de façon complète et correcte le fonctionnement du système [Davis et Hamscher, 1988]. La complétude et l'absence d'erreurs dans le modèle, font partie des hypothèses premières de ces approches de diagnostic. La proposition que nous faisons dans nos travaux vise justement à lever cette hypothèse qui conditionne les résultats de tous les travaux cités ci-dessus.

1.6.3 Les approches à base de données

Finalement, la troisième catégorie de méthodes de diagnostic est celle des méthodes travaillant à partir des historiques d'observations, dites méthodes à base de données. Le modèle est construit à partir de la collecte de données lors du fonctionnement du système. Le modèle reconstruit du procédé met en relation les données collectées avec les différents paramètres du procédé notamment les entrées et sorties. Ces approches supposent que la connaissance disponible sur le système se limite à son observation passée et présente. L'avantage de ce type d'approche, c'est que la construction du modèle ne nécessite donc pas de connaissances profondes sur le système. Parmi ces approches se trouvent les méthodes statistiques [kempowsky, 2004], de reconnaissance des formes [Isaza Narvaes *et al.*, 2006] et également les réseaux de neurones [Hernandez De Leon, 2006].

Nos travaux de recherche se situent essentiellement dans le domaine du diagnostic à base de modèles et portent plus particulièrement sur l'utilisation d'un modèle de bon fonctionnement. Nous reviendrons plus longuement sur les principes d'une telle approche et sur les principaux travaux dans ce domaine dans le chapitre suivant.

Avant cela, nous allons nous intéresser aux outils permettant d'obtenir le modèle qui sera pris comme référence du fonctionnement correct du système.

1.7 Outils de modélisation des Systèmes à Evénements Discrets (SED)

Nous commençons cette section par une brève introduction sur les Systèmes à Evénements Discrets (SED) dans la mesure où les systèmes manufacturiers auxquels nous nous intéressons font partie des SED.

1.7.1 Introduction aux SED

Plusieurs types de systèmes peuvent être spécifiés par des modèles à événements discrets parmi lesquels, les systèmes de transport (aérien, ferroviaire), les systèmes de communication, les web-services, les systèmes de production [Ho, 1989]. Tous sont des exemples de systèmes dynamiques dont l'activité est conditionnée par l'occurrence d'événements à certains instants du temps tels que la fin d'exécution d'une tâche, l'arrivée d'un produit, etc.

Définition des SED

Les Systèmes à Evénements Discrets (SED) sont des systèmes dynamiques qui évoluent en accord avec l'occurrence abrupte d'événements à des intervalles irréguliers [Ramadge et Wonham, 1989]. Ces systèmes ont deux propriétés [Cassandras, 1993] :

1. l'espace d'état est décrit par un ensemble discret ;
2. les transitions d'états sont données par l'occurrence d'événements. Certaines occurrences de ces événements peuvent être identifiées comme des actions spécifiques provoquées (appui sur un bouton) et d'autres comme des occurrences spontanées dictées par la nature (panne intempestive) [Cassandras et Lafortune, 1999].

Exemple

Considérons l'exemple du stockage de produits finis au sein d'un système de production. Quand un nouveau produit est fini, il arrive à la section de stockage. L'arrivée d'un camion entraîne le chargement d'un produit dans le camion pour sa distribution. Ces deux actions qui respectivement incrémentent et décrémentent le nombre de produits stockés sont considérées comme des événements et sont étiquetées par :

e_1 : arrivée au stockage d'un produit terminé,
e_2 : départ d'un produit terminé de la section de stockage.

Ce système de stockage peut être modélisé par un système à événements discrets. La trajectoire qui donne l'évolution du nombre de produits stockés est montrée sur la figure 1.3.

L'état $x(t)$ représente la quantité de produits dans le stock, il est incrémenté de 1 à chaque événement e_1 et il est décrémenté de 1 à l'occurrence de l'événement e_2. La quantité de produits à l'état initial est supposée être nulle. L'occurrence d'un événement e_1 a lieu aux instants t_1, t_2, t_4 et t_5. L'occurrence de l'événement e_2 est à l'instant t_3.

Lorsque la séquence d'événements d'entrée du SED est spécifiée de manière déterministe (i.e. suite à l'occurrence d'un événement, l'état suivant du système est unique)

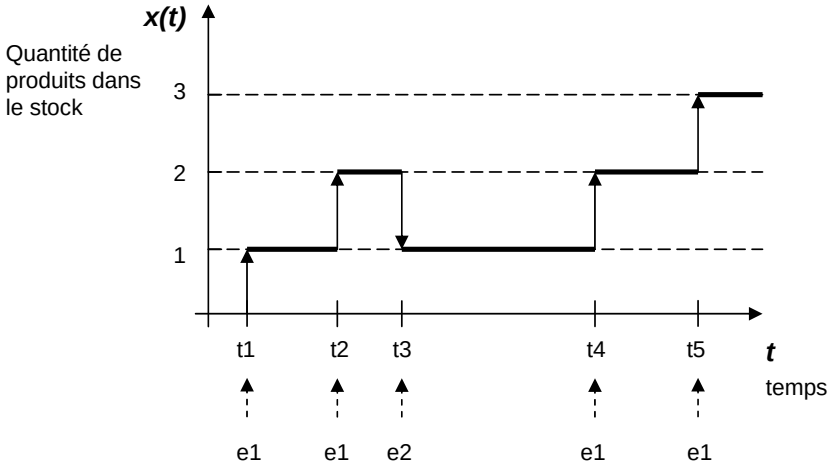


FIG. 1.3 – Trajectoire du système de stockage

comme une séquence d'événements e_1, e_2 sans informations sur les dates d'occurrences des événements, il est possible d'obtenir un modèle de comportement logique. Si des dates d'occurrence peuvent être associées aux événements de la séquence, un modèle temporisé du système peut être construit. Et si des informations sur la distribution de probabilité des événements sont prises en compte alors il est possible de disposer de modèles stochastiques. Ces trois niveaux d'abstraction peuvent être considérés lors de modélisation des SED. Le choix d'un niveau ou d'un autre dépendra fortement des informations disponibles mais également du but de la modélisation et de l'analyse du système. Par exemple un modèle de comportement logique est utilisé dans le cas où un ordre précis de l'occurrence des événements est recherché afin de déterminer si ces événements satisfont un ensemble de spécifications ou si ces événements atteint un état spécifique. Maintenant s'il s'agit de déterminer si l'occurrence de ces événements respecte un certain délai, il est nécessaire de mettre en place des modèles temporisés. Finalement, dans le cas où l'objectif est de déterminer la probabilité qu'un système évolue vers un état précis, les modèles stochastiques sont utilisés.

La section suivante est dédiée aux principaux outils permettant de modéliser le comportement logique d'un système à événements discrets et donne une courte présentation des caractéristiques de chacun.

1.7.2 Les outils de modélisation des SED

1.7.2.1 Les automates

Un des moyens d'étudier le comportement logique d'un SED est basé sur la théorie de langages et les automates. Le point de départ de cette théorie est que à chaque SED est associé un ensemble d'événements. Cet ensemble E peut être vu comme un alphabet d'un langage et les séquences d'événements sont des mots de ce langage [CASSANDRAS et LAFORTUNE, 1999]. Un automate est un dispositif qui génère un langage en utilisant cet alphabet E selon certaines règles bien définies. Cela revient à spécifier des ensembles d'états et des transitions entre ces états.

Définition d'un automate :

Formellement un automate d'état fini est défini par un 5-uplet

$$A = (Q, E, f, q_0, Q_m)$$

où

- Q est l'ensemble fini d'états
- E est un alphabet fini
- f est la fonction transition, $f : Q \times E \rightarrow Q$
- $q0$ est un état initial, $q0 \in Q$
- Q_m est un ensemble d'états finaux, $Q_m \subseteq Q$

L'automate peut être représenté graphiquement par un diagramme de transitions d'états. Il s'agit d'un graphe orienté dans lequel les états sont représentés par des cercles et les transitions par des arcs entre états. Un arc étiqueté par un événement e et connecté à deux états étiquetés q et q', représente une transition de l'état courant q vers l'état q', suite à l'occurrence de l'événement e.

Exemple :

La figure 1.4 montre la représentation graphique d'un automate $A = (Q, E, f, q_0, Q_m)$. L'ensemble des états de cet automate est $Q = \{q_0, q_1, q_2\}$. L'état initial q_0 est indiqué par une flèche entrante. Les états finaux (aussi appelés marqués) sont représentés par des doubles cercles, $Q_m = \{q_0, q_2\}$. L'ensemble des symboles (étiquettes) associés aux arcs du diagramme est l'ensemble d'événements $E = \{a, b, g\}$. Les arcs du diagramme sont la représentation graphique des transitions entres les états de l'automate, $f : Q \times E \rightarrow Q$:

$$f(q_0, a) = q_0, \quad f(q_0, g) = q_2, \quad f(q_1, a) = q_0, \quad f(q_1, b) = q_1, \quad f(q_2, b) = q_2,$$
$$f(q_2, a) = f(q_2, g) = q_1$$

La notation $f(q_0, g) = q_2$ (on utilise aussi couramment la notation (q_0, g, q_2)) signifie que si l'automate est dans l'état q_0, à l'occurrence de l'événement g, l'automate évolue vers l'état q_2. Deux remarques importantes peuvent être faites à partir de cet exemple :

1. un événement peut avoir lieu et ne pas changer l'état de l'automate (i.e. $f(q0, a) = q0$),
2. pour un état donné, deux événements différents peuvent engendrer des transitions vers un même état ($f(q2, a) = f(q2, g) = q1$),

Définition du langage généré (ou reconnu) par un automate

Rappelons tout d'abord la définition de la fermeture de Kleene. La fermeture de Kleene (ou étoile de Kleene) est un opérateur défini par

$$E \stackrel{*}{\rightarrow} E^*$$

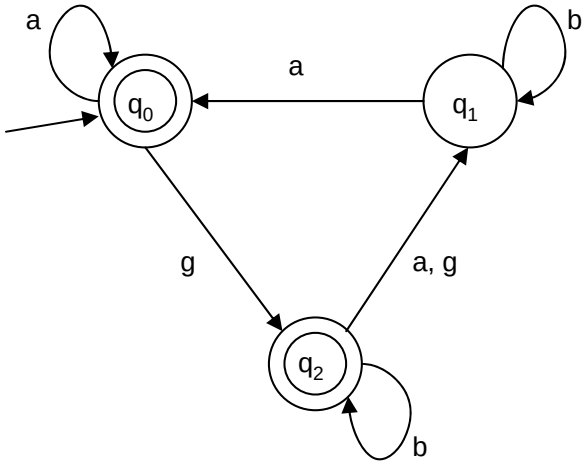


FIG. 1.4 – Diagramme de transitions d'états [CASSANDRAS et LAFORTUNE, 1999]

avec E un ensemble de symboles, c'est à dire un alphabet, et E^* l'ensemble de toutes les séquences de symboles de E incluant la séquence vide notée ϵ, c'est à dire tous les mots pouvant être construits sur l'alphabet E.

Le langage généré par l'automate $A = (Q, E, f, q_0, Q_m)$ est défini par

$$L(A) := \{s \in E^* : f(q0, s) \text{ est défini}\}$$

$L(A) \in E^*$ et représente tous les chemins qui peuvent être suivis dans le diagramme de transitions d'états, à partir de l'état initial.

Notons également que la fermeture de Kleene est également définie pour un langage comme étant l'ensemble des phrases construites avec les mots de ce langage.

D'après [CASSANDRAS, 1993] le langage peut être représenté par une expression compacte appelée expression régulière. Ces expressions régulières sont définies à partir d'opérations élémentaires sur les chaînes de caractères (string). Ces opérations sont : la concaténation, l'intersection, l'union et la fermeture de Kleene. Par exemple soient $L_1 = \{a, b\}$ et $L_2 = \{d\}$ deux langages, la concaténation $L_1L_2 = \{ad, bd\}$ est formée par tous les éléments de L_1 suivis par tous les éléments de L_2. L'union et l'intersection de ces deux langages sont définies comme l'union et l'intersection de leurs ensembles : $L_1 \cup L_2 = \{a, b, d\}$ et $L_1 \cap L_2 = \emptyset$.

Définition du langage marqué

Le langage marqué généré par l'automate $A = (Q, E, f, q0, Q_m)$ est défini par

$$L_m(A) := \{s \in L(A) : f(q0, s) \in Q_m\} \ (Q_m \text{ est dit ensemble des états marqués})$$

$L_m(A)$ est le sous-ensemble de $L(A)$ contenant tous les chemins de $L(A)$ qui conduisent vers un des états marqués.

Un automate est dit déterministe si l'état initial est unique et si la relation de transition, appliquée à un couple (q_i, e) définit toujours un état unique. Les automates qui ne respectent pas cette propriété sont dits non déterministes.

Le plus grand inconvénient de ce type de modélisation est l'explosion combinatoire du nombre d'états quand il s'agit de représenter des systèmes complexes. Pour pallier cette limitation les machines à états ont été étendues donnant lieu notamment aux Statecharts qui ont été par la suite intégrés dans UML (Unified Modeling Langage) pour le modélisation orientée objet. Les réseaux de Petri sont un autre outil de modélisation remarquable dans le domaine des SED permettant de faire face à ce problème d'explosion combinatoire.

1.7.2.2 Les Statecharts

Les Statecharts ont été développés par David Harel in 1983 comme un formalisme pour la spécification de comportements réactifs [Harel, 1988]. Les Statecharts sont des machines à états finis étendues avec des concepts de hiérarchie et de parallélisme qui permettent de représenter un système d'une façon plus compacte et qui évitent l'explosion combinatoire.

Les Statecharts décrivent le comportement du système, c'est-à-dire, comment les composants d'un système communiquent, collaborent et comment ces composants réalisent leurs propres comportements internes. Le comportement du système est décrit au travers de tous les états possibles de ses composants et de la façon selon laquelle les événements agissent sur ces composants. Un statechart est composé :

- **d'états**, ils sont représentés par des boites arrondies. les états peuvent être terminaux ou abstraits. L'état initial est indiqué par l'extrémité d'arcs (dits arcs initiaux) dont l'origine n'est pas un état mais un petit cercle. Un état abstrait peut faire apparaître plusieurs zones séparées par des lignes pointillées. Dans ce cas, chaque zone contient la description d'un statechart et ces différents statecharts évoluent en parallèle. Les synchronisations entre ces états parallèles sont assurées par des événements dont la diffusion est instantanée dans tout le statechart ;
- **de transitions**, elles sont représentées par des arcs orientés qui relient deux états ;
- **d'événements déclencheur**, ils peuvent être externes ou internes au statechart et sont la cause des transitions entre les états. Ils sont représentés comme des étiquettes sur les transitions ;
- **de conditions**, ce sont des expressions booléennes qui qualifient les événements. Ces conditions sont représentées sur les transitions entre des crochets comme suit : événement[condition] ;
- **d'événements générés**, qui sont diffusés lors du franchissement d'un arc. Ils sont représentées sur les transitions avec l'événement et la condition, juste après le symbole " / " . La forme générale de l'étiquette d'un arc dans un statechart est donc $e_i[c_j]/e_o$ avec e_i événement déclencheur, c_j condition et e_o événement généré.

Le mécanisme de diffusion assure l'évolution synchrone du statechart puisque tout événement produit à un instant t est vu par toutes les transitions au même instant t et provoque donc d'éventuelles évolutions instantanément. Intéressant dans sa forme graphique, en particulier par sa capacité d'abstraction de comportements, cet outil ne dispose pas d'un formalisme mathématique permettant de faire des preuves formelles. Il faut donc se ramener à l'automate équivalent et le problème de l'explosion combinatoire du nombre d'état réapparaît.

La figure 1.5 donne un exemple de la représentation graphique d'un Statechart. Dans ce diagramme l'état s0 est décomposé en un automate à trois états s_1, s_2, s_3. L'état s_1 est lui-même décomposé en deux états s_4 et s_5 qui à leur tour sont décomposés respectivement en s_6, s_7 et s_8, s_9. Les états s_0, s_1, s_4 et s_5 sont appelés états composites. Les transitions sont étiquetées par des événements/ actions comme r_1/a_1.

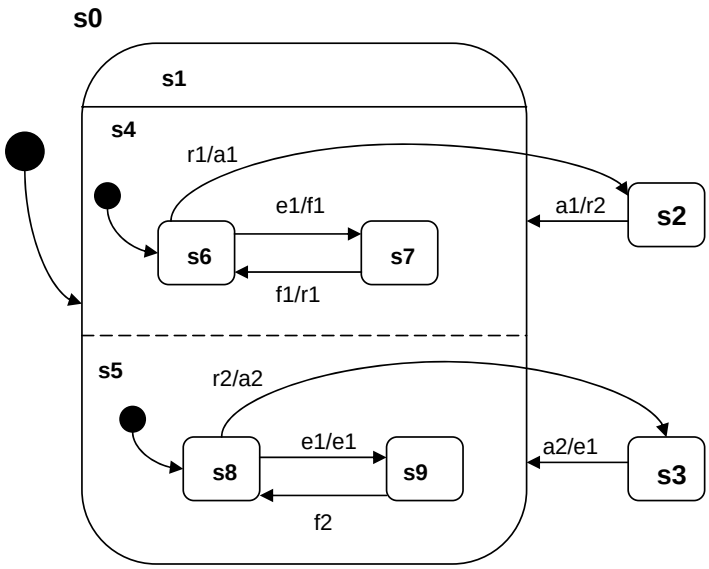


FIG. 1.5 – Diagramme Statecharts

L'état du système est modélisé par des configurations c'est à dire par des ensembles d'états. Par exemple dans la figure 1.5 le système peut être dans les configurations suivantes : $\{s_1, s_6, s_8\}$, $\{s_1, s_6, s_9\}$, $\{s_1, s_7, s_8\}$, $\{s_1, s_7, s_9\}$, $\{s_2\}$, $\{s_3\}$. Une transition est sensibilisée et peut être tirée si et seulement si son état de départ fait partie de l'état courant du système. Par exemple, si le système se trouve dans la configuration $\{s1, s6, s9\}$ la transition étiquetée par $r1/a1$ peut être tirée, alors l'état de départ sera quitté et l'état $\{s2\}$ sera atteint.

Après 1994, les statecharts ont été, comme d'autres outils et méthodes, intégrés au système de modélisation orienté objet UML (Unified Modeling Langage) pour doter ce dernier d'un modèle exécutable et d'une analyse dynamique [MAGEE et DE WECK, 2004]. Cette modélisation orientée objet (UML) a été utilisée d'abord dans le domaine de l'informatique où elle permet de modéliser informatiquement un ensemble d'éléments d'une partie du monde réel (appelée domaine) en un ensemble d'entités informatiques, appelées objets. Récemment cette technique à pris beaucoup d'importance dans le domaine de la modélisation de systèmes complexes en général. Les Statecharts y ont un rôle important pour la modélisation du comportement des systèmes.

1.7.2.3 Les réseaux de Petri

Les réseaux de Petri (RdP) ont été développés par le mathématicien Allemand Carl Adam Petri en 1962. Au début, cet outil permettait de décrire des relations existant entre des conditions et des événements. Par la suite, les RdP ont fait l'objet d'enrichis-

sements portant sur tous les aspects liés à la modélisation des comportements parallèles et distribués [PETERSON, 1981].

Graphiquement, un réseau de Petri est un graphe qui comporte deux types de noeuds : les places et les transitions. Une place est représentée par un cercle et une transition est représentée par un trait ou un rectangle dans certains des cas. Les places et les transitions sont reliées par des arcs orientés allant d'une place à une transition ou d'une transition vers une place. Chaque place contient un nombre entier (positif ou nul) de marques ou jetons. Le marquage des places à un certain instant définit l'état du RdP et ainsi l'état du système décrit par le RdP. L'évolution de l'état correspond à une évolution du marquage, qui se produit par le franchissement des transitions.

Le franchissement d'une transition peut s'effectuer seulement si les places en amont de cette transition contiennent au moins un nombre de jetons égal au poids de l'arc reliant la place à la transition. Le franchissement de la transition consiste à retirer des places amont, une quantité de jetons égale au poids de l'arc entrant dans la transition, et à ajouter un nombre de jetons égal au poids de l'arc sortant de la transition dans les places en aval de la transition.

Les définitions formelles sont données ci-dessous :

Définition d'un RdP

Un réseau de Petri peut être défini par un 4-uplet R=(P, T, Pre, Post) , où :

- P est l'ensemble fini de places,
- T est l'ensemble fini de transitions
- Pre : $P \times T \rightarrow N$ est l'application places précédentes qui définit les arcs allant des places vers les transitions ;
- Post : $P \times T \rightarrow N$ est l'application places suivantes qui définit les arcs des transitions vers les places ;

Pre(p,t) est le poids de l'arc de la place p à la transition t ; Post(p,t) est le poids de l'arc de la transition t à la place p ;

Définition d'un RdP marqué

Un réseau de Petri marqué est défini comme un couple (R, m) dans lequel R est un réseau de Petri et $m : P \rightarrow N$ est une application appelée marquage.

Dans le RdP une transition est dite franchissable (ou encore sensibilisée) si :

$$\forall p \in P \qquad M(p) \geq Pre(p,t)$$

où M(p) dénote le nombre de jetons que contient la place p. L'évolution du marquage d'une place p lors du tir d'une transition t est calculée par l'équation fondamentale d'un RdP :

$$M'(p) = M(p) + C(p,t)$$

où C est la fonction d'incidence du RdP définie par

$$P \times T \stackrel{C}{\rightarrow} N$$
$$C(p,t) = Post(p,t) - Pre(p,t)$$

et représente la variation du marquage induit sur la place p par le tir de la transition t.

Sous la forme matricielle, cette équation se généralise à

$$M' = M + C.\overline{s}$$

où $\overline{s}$ est un vecteur de $N^{|T|}$ dans lequel la composante s_i indique le nombre de franchissements de la transition t_i dans la séquence considérée.

Exemple

Considérons le réseau de Petri représenté par la figure 1.6. Ce réseau est formé par l'ensemble des places $P = \{p_1, p_2, p_3, p_4, p_5\}$. L'ensemble de transitions est $T = \{t_1, t_2, t_3, t_4\}$. Supposons que le marquage courant du réseau soit $M_2 = [01100]^T$. La i^{eme} composante de ce vecteur colonne (par simplicité nous écrivons ce vecteur sous sa forme transposée) est le nombre de jetons associé à la place p_i. Ainsi les places p_2 et p_3 contiennent chacune un jeton. Deux transitions sont sensibilisées par ce marquage $M_2 : t_2$ et t_3. De ce fait, la prochaine évolution du réseau peut correspondre au franchissement de t_2 ou à celui de t_3, mais aucun autre franchissement de transition n'est possible depuis cet état du réseau. Le franchissement de t_3 conduit au marquage $M_3 = [01001]^T$, celui de t_2 conduit vers l'état décrit par $M_4 = [00110]^T$. Pour que la transition t_4 soit franchissable il est nécessaire d'avoir le marquage $M_5 = [00011]^T$ ce qui signifie que les transitions t_2 et t_3 doivent avoir être franchies auparavant. Après le franchissement de t_4 le réseau retrouvera son marquage initial $M_0 = [10000]^T$.

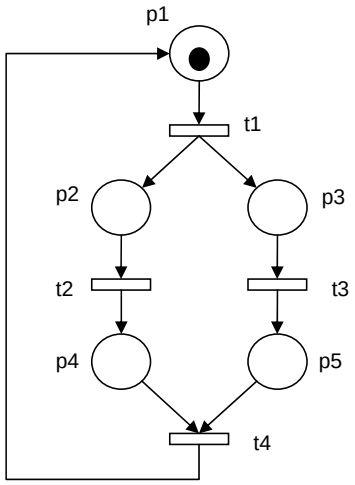


FIG. 1.6 – RdP marqué

Les réseaux de Petri permettent de modéliser différentes situations présentes dans les systèmes comme le partage de ressources, les synchronisations d'activités, des activités concurrentes ou parallèles, des activités exclusives, des liens de causalités etc.

Il existe de nombreuses familles de réseaux de Petri certaines offrant par rapport au modèle de base, des enrichissement sur les données (RdP colorés, RdP Prédicats-Transitions, RdP à Objets, RdP à marquage flou,...), d'autres des enrichissements sur le temps (RdP temporisés, temporels, ...), d'autres encore introduisent les aspects stochastiques. Les réseaux de Petri sont donc un outil de modélisation disposant d'un pouvoir

d'expression extrêmement riche. La puissance de cet outil de modélisation est renforcée par l'existence de différentes techniques d'analyse qui lui sont associées. Ces techniques permettent d'analyser les propriétés d'un réseau de Petri comme les propriétés de vivacité, bornitude, réinitialisabilité mais également des propriétés structurelles [Diaz, 2001] [David et Halla, 1989] [Peterson, 1981].

Nous allons terminer ce chapitre introductif en présentant le domaine d'application de nos travaux que sont les systèmes de production manufacturière.

1.8 Domaine d'application

1.8.1 Les Systèmes Flexibles de Production Manufacturière

L'avancée technologique, l'ouverture de l'entreprise aux marchés internationaux, l'augmentation des exigences des clients font partie des raisons de l'évolution des systèmes de production traditionnels [Pinto-Leitão, 2004]. Pour s'adapter aux nouvelles exigences ces systèmes ont dû introduire de nouvelles technologies leur permettant d'atteindre le niveau de flexibilité voulu [Besant, 1989]. La flexibilité permet d'augmenter et de varier la production, de réduire les cycles de production, d'augmenter la qualité des produits, etc. Cette flexibilité se décline à différents niveaux [Ausfelder *et al.*, 1993] :

- **flexibilité physique**, qui se traduit par la capacité d'adaptation des ressources qui constituent l'atelier. En d'autres termes dans la possibilité de changement dans l'ordre d'opérations ou le remplacement des opérations d'un ressource (polyvalence) ;
- **flexibilité dans le système de transport**, qui traduit la richesse de connexions entre les différents moyens de production ;
- **flexibilité dans les processus**, indique la capacité du système à produire un ensemble de produits différents sans perdre du temps au niveau des phases de réglage ;
- **flexibilité du système**, qui décrit la gamme et la quantité de produits que le système est capable de produire dans une certaine période de temps.

La flexibilité a augmenté la productivité, mais elle a également augmenté la complexité structurelle et opérationnelle des systèmes de production [Kuzgunkaya et Elmaraghy, 2006]. Depuis son début en 1960, la flexibilité a été synonyme d'automatisation [Pinto-Leitão, 2004]. A partir de là, des anciennes technologies ont été remplacées par des technologies automatisées comme les *Automates Programmables Industriels* (API), des robots industriels, machines de contrôle numérique, véhicules automatiquement guidés, systèmes automatiques de récupération de matériel et stockages, etc. Peu à peu les Systèmes Flexibles de Production Manufacturière (SFPM) sont devenus des systèmes automatisés de production manufacturière [Niel et Craye, 2002].

De façon générale, un système automatisé est organisé en différents sous-ensembles fonctionnels qui assurent chacun des fonctions élémentaires du système. C'est l'asso-

ciation de ces fonctions qui conduit à l'obtention de la valeur ajoutée des produits. Ainsi un SFPM vu comme un système automatisé est formé par les composants suivants [Samad *et al.*, 2007] :

- **les capteurs** qui prélèvent des informations sur l'état du système ou de son environnement et les codent afin de les rendre exploitables par le système de commande ;
- **les préactionneurs** qui reçoivent des ordres en provenance du système de commande et provoquent la distribution d'énergie vers les actionneurs ;
- **les actionneurs** qui convertissent l'énergie reçue en énergie mécanique le plus souvent ;
- **les effecteurs** qui sont associés aux actionneurs et qui agissent sur le produit pour lui conférer sa valeur ajoutée ;
- **les éléments de la commande** qui traitent l'ensemble des informations afin de gérer le fonctionnement du système ;
- **les éléments de dialogue** qui assurent l'échange d'informations entre l'opérateur et le système ;
- **les éléments de communication** qui assurent l'échange d'informations entre le système et d'autres systèmes.

La figure 1.7 montre les interactions entre ces différents composants. L'accomplissement des fonctions de ces composants permet de transformer une pièce brute en produit terminé. Cette transformation est réalisée par une séquence opérations dictée par le système de commande à l'ensemble des moyens de production qui doivent exécuter ces opérations. Cette séquence d'opérations lancée par la commande traduit les consignes à respecter comme les délais de fabrication, qualité des produit, dimensions, quantité, etc.).

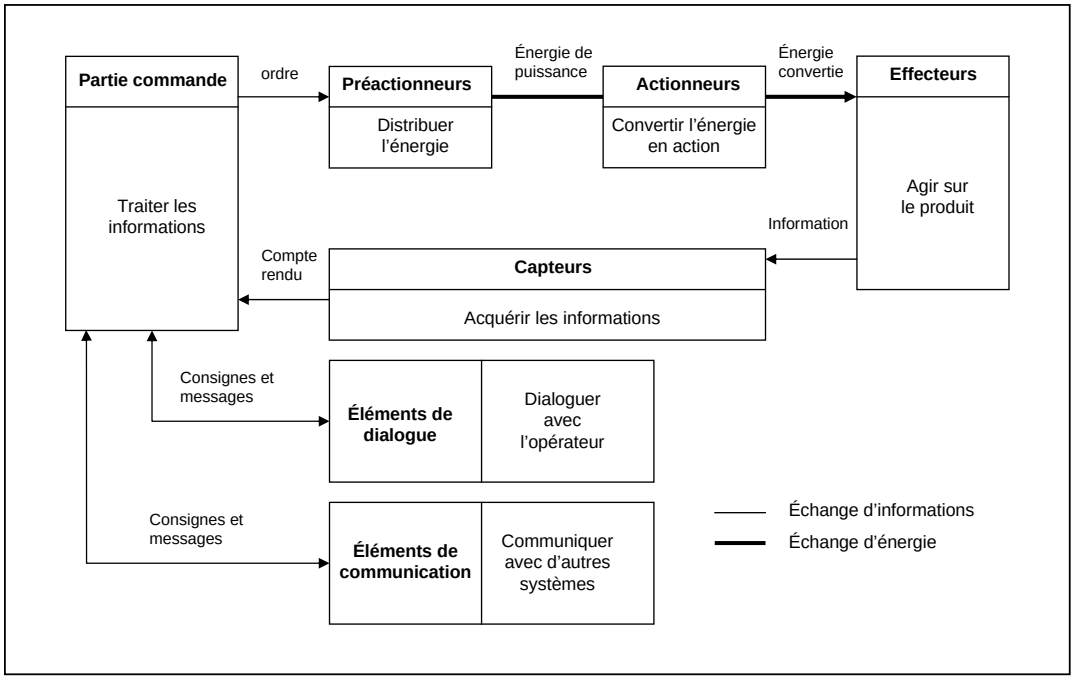


Fig. 1.7 – Interactions entre les composants d'automatisation

Cette évolution des systèmes a entraîné une explosion du nombre de décisions à prendre relatives à la commande de ces systèmes et donc une grande complexité de leur commande [Babiceanu *et al.*, 2004]. De nouvelles architectures de commande ont alors été proposées [Dilts *et al.*, 1991] [Duffie et Prabhu, 1996] [Shiming *et al.*, 2000] comme nous allons le présenter maintenant.

1.8.2 Architectures de commande des systèmes flexibles manufacturiers

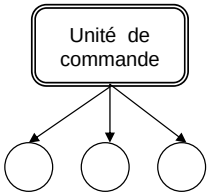
Comme nous l'avons déjà mentionné, dans un système de production flexible, c'est le système de commande qui coordonne et organise les activités de production transformant les pièces en produits finis, ainsi que le transfert de ces pièces entre les différentes étapes de leur transformation. Lors de la transformation d'un produit la commande fixe les types de ressources à utiliser, la séquence d'activités, les délais attendus pour chaque activité, etc. Une architecture de commande décrit comment les différentes fonctions du système de commande sont réparties entre les éléments qui composent ce système. Autrement dit, l'architecture décrit l'organisation du système de commande à partir des composants dédiés à la commande [Dilts *et al.*, 1991] [Kramer et Senehi, 1993] (i.e. API, ordinateurs, panneau de contrôle, etc). L'efficacité des décisions prises par le système de commande pour satisfaire les requêtes de production et optimiser les performances du système dépend fortement de l'architecture [Benielli et Cerato, 2001].

Quatre types d'architecture peuvent être considérés [Dilts *et al.*, 1991] : architecture centralisée, architecture hiérarchique, hiérarchique modifiée et hétérarchique. Nous décrivons par la suite les principales caractéristiques de ces quatre architectures.

1.8.2.1 Les architectures centralisées

Les premiers types d'architecture utilisés pour organiser un système de commande ont été les architectures dites centralisées. La notion de centralisation est utilisée pour indiquer qu'une seule unité de commande est responsable de toute prise de décision dans le système de commande et pour mentionner la concentration de l'information au sein de cette seule unité. Cette unité organise et coordonne tout ce qui se réfère à la production (planification, ordonnancement, transfert de matériaux,...) par l'envoi des ordres aux contrôleurs des ressources. C'est également cette unité qui reçoit les informations en provenance des capteurs du procédé et des contrôleurs des ressources pour ensuite les utiliser dans la prise de décision globale. La figure 1.8 représente une architecture centralisée et donne les principaux avantages et inconvénients d'une telle architecture [Babiceanu *et al.*, 2004].

Compte tenu de la complexité des systèmes de production, des avancées technologiques dans le domaine et de la faible capacité de ces architectures à s'adapter, de nouvelles architectures appelées hiérarchiques ont été conçues. Dans ces architectures le système de commande suit une décomposition par niveaux.



Avantages :

•Moins d'éléments sont nécessaires pour organiser et coordonner le système production.

•L'information sur l'état global du système est concentrée dans une seule unité.

•Accès à l'information globale complète qui facilite l'optimisation de certains aspects de la production

Désavantages :

•Temps de réponse très long quand le système est composé d'un grand nombre de ressources.

•Très peu tolérante aux défaillances, car si l'unité de contrôle tombe en panne tout le système tombe en panne.

•Peu de possibilités de modifications et d'extensions car très dépendant de la capacité de l'unité centrale de commande.

•Difficulté pour les modifications dans le software car la logique est cachée dans la structure de données globale.

FIG. 1.8 – Architecture centralisée

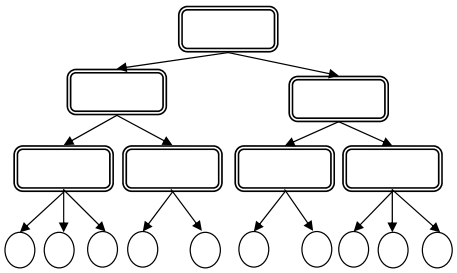
1.8.2.2 Les architectures hiérarchiques

Ce type d'architecture est caractérisé par une forme pyramidale où les différents niveaux ont leurs propres objectifs et fonctions. L'architecture est composée par différentes unités de contrôle organisées en différents niveaux. Toutes les activités de contrôle d'un niveau inférieur sont dictées par le niveau supérieur. Le niveau plus haut de la hiérarchie est composé d'une seule unité de contrôle qui formule les objectifs et les stratégies globales. Les décisions de contrôle agrégées suivent un flux descendant dans la structure. Ces décisions sont plus affinées et détaillées dans les niveaux bas de la hiérarchie. Les informations relatives aux opérations réalisées par le système (i. e. comptes rendus) suivent un flux ascendant dans la structure. Les relations entre un niveau supérieur et un niveau inférieur ont été définies comme des relations de type maître/esclave. Les principales caractéristiques de ce type d'architecture sont présentées dans la figure 1.9 [DUFFIE, 1990] [DILTS *et al.*, 1991].

Etant donné le manque d'autonomie des niveaux subordonnés, le temps requis pour la prise de décision dans le cas de certaines défaillances, les coûts élevés pour la maintenance et la tolérance aux défaillances dans l'architecture, ces architectures ont été modifiées afin d'autoriser les communications entre les entités d'un même niveau, leur donnant ainsi un peu plus d'autonomie comme nous allons le voir dans la partie suivante.

1.8.2.3 Les architectures hiérarchiques modifiées

L'architecture modifiée est dérivée de l'architecture hiérarchique avec laquelle elle partage plusieurs caractéristiques comme l'existence des niveaux de contrôle. Mais les relations maître/esclave entre ces niveaux sont remplacées par des relations superviseur/subordonné. Les niveaux subordonnés ont un degré d'autonomie plus élevé dans la mesure où il existe un degré minimal de communication et de coordinations entre les subor-



Avantages:

•La relation maître/esclave diminue le temps de réponse par rapport à l'architecture centralisée.

•Les relations entre les unités de contrôle adjacentes sont faciles à comprendre.

•La redondance peut être incorporée en intégrant d'autres contrôleurs à l'architecture qui sont utilisés dans le cas de la défaillance d'une unité de contrôle (tolérance aux défaillances).

•Elles présentent la possibilité d'obtenir l'information globale sur le système au travers des niveaux supérieurs.

Désavantages :

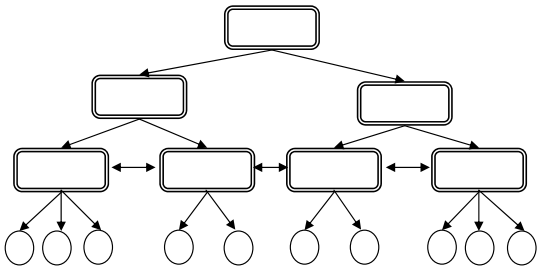
•Les modifications de software et hardware non prévues affectent tous les niveaux de la hiérarchie à cause de ses interdépendances.

•Les décisions globales sont basées sur des informations obsolètes du système à cause des délais dans l'agrégation de l'information

•La défaillance d'une unité de contrôle paralyse toutes les unités des niveaux inferieurs qui dépendent d'elle.

•Les relations entre les niveaux de contrôle demandent une connaissance substantielle du niveau supérieur et du niveau inferieur pour la tolérance aux défaillances.

FIG. 1.9 – Architecture hiérarchique



Avantages:

•Niveaux subordonnés plus autonomes lesquels n'ont pas besoin d'une supervision permanente.

•Temps de réponse plus court du superviseur: moins de taches ordinaires sont exécutées par le superviseur, ce qui lui permet de répondre plus rapidement.

•Diminution du problème d'arrêt des niveaux dépendants dans le cas d'une défaillance dans le niveau supérieur.

Désavantages :

•Problèmes de connectivité et communication entre les niveaux subordonnés.

•Limitations des contrôleurs des niveaux inferieurs, étant donné sa faible sophistication.

•Incrément de la difficulté de conception du système de commande à cause des interactions entre les subordonnés .

FIG. 1.10 – Architecture hiérarchique modifiée

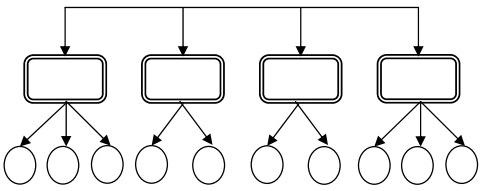
donnés pour réaliser une séquence d'activités initiée par une commande du superviseur. Ce type d'architecture est aussi appelé architecture décentralisée [DILTS *et al.*, 1991] [BABICEANU *et al.*, 2004]. La figure 1.10 fait le bilan des caractéristiques de cette architecture.

La limitation principale de ce type d'architecture reste l'autonomie totale des niveaux subordonnés. Cette limitation est en fait présente dans les architectures centralisées, hiérarchiques et hiérarchiques modifiées. L'ensemble des problèmes non résolus avec ces

types d'architectures a donné lieu à une architecture dite hétérarchique ou distribuée.

1.8.2.4 Les architectures hétérarchiques

Dans les structures de commande hétérarchiques la notion de niveaux a été remplacée par un seul niveau composé d'entités de contrôle locales indépendantes. Ces entités ont pleine autonomie dans leur contexte local, elles coopèrent et communiquent avec les autres entités de l'architecture. La notion de maître/esclave, où il existe une entité qui commande et une autre qui obéit a disparu. Ici les relations de coopération entre les entités sont gérées par processus de négociation où une entité de contrôle envoie une requête vers une autre entité qui peut décider d'accepter ou de refuser cette requête en fonction de ses propres connaissances locales [Hatvany, 1985]. Cette dernière caractéristique assure la notion d'autonomie locale dans la phase de coopération. La notion de distribution implique la disparition de l'information globale, ainsi seules les informations locales sont conservées dans cette architecture [Dilts *et al.*, 1991] [Duffie *et al.*, 1988] [Leitao et Restivo, 1999]. Ces architectures ont résolu un certain nombre des problèmes présents dans les autres types d'architectures, mais les notions de distribution et d'autonomie ont donné naissance à de nouveaux problèmes comme le montre la figure 1.11.



Avantages:

•La modularité réduit la complexité, favorise le développement et la maintenance du software.

•Augmentation de la tolérance aux défaillances car la défaillance dans une entité n'affecte pas tout le système.

•La possibilité de modifier les protocoles de coopération pour la prise de décision favorise la reconfigurabilité et l'adaptabilité.

•Rapidité dans la diffusion de l'information.

Désavantages :

•Incompatibilité des systèmes d'exploitation des différents modules.

•Problèmes dans la capacité du réseau de communication.

•Différence entre les protocoles de communication.

• Difficulté pour le fonctionnement d'un contrôle coopératif.

Fig. 1.11 – Architecture hétérarchique

Pour compenser certaines lacunes des architectures traditionnelles hiérarchiques et hétérarchiques, de nouvelles architectures ont émergé, telles les architectures fractales, bioniques, holoniques qui se basent sur des analogies avec des systèmes d'organisation sociale. Ces nouvelles architectures, que nous ne détaillerons pas ici, sont à rapprocher des architectures distribuées avec des entités autonomes qui opèrent comme un ensemble d'entités coopérantes [Babiceanu *et al.*, 2004] [Tharumarajah *et al.*, 1996] [McFarlane *et al.*, 1995] [Tharumaraja *et al.*, 1998].

Ces architectures de contrôle ont été introduites dans ce chapitre avec l'objectif de mettre en évidence l'influence que ces architectures ont sur les architectures de diagnostic présentées dans le chapitre 2. Ces architectures de diagnostic sont en effet généralement calquées sur les architectures de contrôle.

Conclusions

Nous avons présenté dans ce chapitre la problématique générale de la commande et surveillance-supervision dans le domaine des systèmes à événements discrets. Dans ce contexte nous nous intéressons au problème de détection et diagnostic de défaillances que nous présentons dans la suite de ce document (chapitres 4 et 6). Au travers de ce premier chapitre, nous avons donné un aperçu général des techniques de diagnostic pour pouvoir par la suite mieux positionner notre approche. Nous avons également introduit les Systèmes Flexibles de Production Manufacturière qui constituent le domaine d'application de nos recherches. La façon dont ont évolué les architectures qui contrôlent ces systèmes est aussi abordée dans ce chapitre afin de montrer l'influence que ces architectures ont eu sur le développement des architectures de diagnostic présentées dans le chapitre 2.

Dans le chapitre suivant nous allons développer la technique du diagnostic à base de modèle sur laquelle porte notre travail de recherche et présenter notre contribution dans ce domaine.

Chapitre 2

Diagnostic à base de modèles pour les systèmes à événements discrets

Introduction

Ce chapitre présente le cadre général de notre travail. Nous introduirons tout d'abord les techniques de diagnostic à base de modèles sur lesquelles s'appuient nos travaux. Notre but étant de positionner nos travaux par rapport aux principales approches existantes, nous ne prétendons pas être exhaustifs dans cette présentation. Classiquement, deux types de modèles peuvent être utilisés : les modèles de bon comportement et les modèles de fautes. Nous détaillerons les principes et le fonctionnement du diagnostic à base de modèles dans ces deux cas.

Suite à la présentation générale du diagnostic à base de modèles nous décrivons ensuite la façon dont cette technique a été utilisée pour le diagnostic de défaillances dans les systèmes à événements discrets. L'hypothèse d'un modèle correct retenue depuis le début de l'approche de diagnostic à base de modèles a donné lieu à nos travaux de recherche, car nous pensons que ne pas considérer la possibilité d'erreurs dans le modèle peut affecter la phase de détection. En effet, ces erreurs peuvent provoquer la détection de faux symptômes causés par une erreur du modèle. Ces erreurs peuvent aussi masquer de vrais symptômes causés par les dysfonctionnements du système.

2.1 Le principe de diagnostic à base de modèles (DBM)

Le Diagnostic à Base de Modèles (DBM), appelé diagnostic à partir des principes premiers, a été introduit par la communauté de l'Intelligence Artificielle (IA) aux Etats-Unis au milieu des années 70 et a été formalisé au début des années 80 dans [Reiter, 1987]. Dans cette technique, l'élément central pour diagnostiquer un système est un modèle de ce système. La notion de modèle est employée ici en opposition à la notion de connaissance généralement empirique utilisée traditionnellement dans les approches de diagnostic à base de règles et systèmes experts, issues de la communauté IA.

De telles approches utilisent des connaissances structurelles ou fonctionnelles du système et ne reposent pas sur l'expérience des experts qui associent les symptômes des

défaillances avec les composants en utilisant leur expérience et la connaissance des dysfonctionnements.

Le principe du diagnostic à base de modèle consiste à confronter le comportement du système observé (par l'intermédiaire des capteurs du système) avec le comportement émulé dans le modèle pour déterminer l'existence des symptômes (détection) et localiser les composants responsables des dysfonctionnements (diagnostic). Des travaux fondamentaux dans ce domaine sont présentés dans [Davis et Hamscher, 1988]. Le DBM traite donc principalement la localisation des défauts mais, après une extension, peut aussi procéder à l'identification de ces défauts.

La mise en place des modèles se place dans le cadre classique de la modélisation orientée composant car les composants objet du diagnostic sont les instances physiques porteuses de défauts potentiels et objets des actions de réparation ou de changement qui seront réalisés.

2.2 Les types de modèles utilisés

Deux axes ont émergé dans le raisonnement à base de modèles pour le diagnostic. La différence entre ces deux axes est le type du comportement décrit par le modèle. Il peut s'agir du comportement nominal du système (ce que le système est supposé faire) ou bien il peut s'agir du comportement que suit le système en présence de défaillances. Il y a donc des approches de DBM dites à base de modèles de bon comportement et d'autres dites à base de modèles de fautes.

2.2.1 Modèle de bon comportement

Le modèle doit capturer la connaissance structurelle et comportementale du système. La connaissance comportementale donne le comportement dynamique et statique de chaque composant en fonctionnement correct. La connaissance structurelle est modélisée en terme d'interactions et de connexions entre les composants.

Ces connaissances doivent permettre de calculer le comportement global du système en réponse à une sollicitation quelconque en fonctionnement normal. Ainsi le comportement décrit dans le modèle devient un comportement de référence, souvent qualifié maladroitement de "comportement prédit" car il peut être déterminé hors ligne et utilisable pendant la phase d'exploitation du système.

Le comportement d'un système est donné par l'ensemble de trajectoires (états et transitions) qui montrent les évolutions du système provoquées par les événements.

Le principe du diagnostic avec modèle de bon comportement

Dans cette approche il n'est pas nécessaire de posséder la connaissance a priori des défaillances pouvant affecter le système. Les mêmes sollicitations (entrées) sont appliquées au système et au modèle et les signaux issus des capteurs du système sont utilisés pour en reconstituer le comportement réel. La détection repose alors sur la constatation d'une incohérence entre le comportement de référence issu du modèle et celui qui est observé sur le système. L'hypothèse fondamentale dans cette approche est que "si le modèle est correct, toutes les différences entre les observations et la référence sont dues aux fautes dans les composants du système". Il s'agit donc d'une détection basée sur la

cohérence. Après la détection d'un symptôme l'étape de diagnostic consiste à localiser les composants à l'origine de ce symptôme. Trois étapes sont suivies pour localiser ces composants.

1. **Détection de conflits**. La détection des conflits constitue la première phase de diagnostic dans cette approche. Il s'agit de déterminer l'ensemble de composants qui peuvent causer un symptôme. Ces composants doivent être liés au symptôme et l'ensemble de ces composants est appelé conflit. La notion de conflit apparaît car dans cet ensemble au moins un composant doit être fautif pour expliquer l'apparition et la détection du symptôme. La quantité de conflits augmente au fur et à mesure de la réception des observations, ce qui permet progressivement d'affiner la localisation des défauts.
2. **Génération et test d'hypothèses**. La seconde phase consiste à engendrer des hypothèses sur les conflits. Cela signifie changer l'hypothèse de fonctionnement correct de certains composants par une hypothèse de dysfonctionnement, de manière à ce que toutes les contradictions disparaissent, c'est à dire de manière à ce que le modèle muni de l'hypothèse de faute reproduise le symptome observé. Un diagnostic est alors défini comme un ensemble de composants fautifs rétablissant la cohérence avec les observations. Après la génération de ces hypothèses, le test consiste à déterminer parmi les composants (déterminés par la génération d'hypothèses) ceux qui expliquent toutes les observations. Par le principe de parcimonie utilisé dans cette approche, en général seuls les diagnostics minimaux sont recherchés [Reiter, 1987]. Des travaux consacrés à établir un mécanisme pour la génération et le test d'hypothèses ont été proposés par [De Kleer et Williams, 1987], le mécanisme résultant est nommé GDE (General Diagnostic Engine).
3. **Discrimination d'hypothèses**. La troisième phase consiste à collecter de l'information additionnelle pour mieux discriminer les diagnostics restants. Il y a deux façons d'obtenir de nouvelles informations [Davis et Hamscher, 1988] : en faisant de nouvelles mesures sur le système s'il reste encore des points à mesurer ou en changeant les entrées du système pour se placer dans une nouvelle configuration et faire ensuite des mesures dans cette configuration.

Un des avantages de cette approche est qu'il n'est pas nécessaire de disposer de la connaissance a priori des défaillances pouvant affecter le système. La détection des symptômes est réalisée simplement par l'apparition d'un comportement hors modèle de bon fonctionnement. Un diagnostic consiste en une phrase logique décrivant un état possible du système, où l'état est une assignation de statuts de normalité (non fautif) ou d'anormalité (fautif) à chaque composant rétablissant la cohérence entre le modèle et les observations [De Kleer, 2003]. La formalisation de cette approche par logique de premier ordre peut être consultée dans [Reiter, 1987].

Depuis la création du DBM est également apparue la question de savoir si le modèle représente correctement le système [Davis et Hamscher, 1988] :

" L'hypothèse d'un modèle correct du système n'est pas vraie dans tous les cas car le modèle est seulement une approximation du système. Il existe toujours des choses que le modèle ne capture pas et en théorie un modèle sera toujours incomplet voir incorrect. "

N'ayant aucun moyen de valider la complétude et d'affirmer qu'un modèle est correct, l'hypothèse d'un modèle correct du système est faite a priori et tous les résultats y sont liés. Cette remarque trouve ici sa place dans le domaine du diagnostic, mais s'applique dans tous les domaines utilisant un modèle : conception, commande, etc.

2.2.2 Modèle de fautes

La technique du diagnostic à base de modèles peut également s'appuyer sur un modèle de fautes. De tels modèles sont généralement nécessaires afin d'identifier les défauts après les avoir localisés. Ils sont aussi le seul moyen de capturer une connaissance formelle sur un mode de dysfonctionnement connu, dans le prolongement des études de fiabilité comme l'AMDE [Villemeur, 1988] [Lopez Varela *et al.*, 2005]. Un modèle de fautes capture la connaissance sur les dysfonctionnements ou défauts pouvant survenir dans le système à diagnostiquer. C'est-à-dire qu'en en plus du comportement correct de chaque composant, sont pris en compte les modes de défaillances des composants. Chaque mode capture une faute élémentaire du composant. Un mode inconnu est ajouté pour tenir compte l'impossibilité d'une énumération exhaustive de tous les défauts possibles. Le comportement de chaque composant est caractérisé pour décrire son comportement dans chacun de ses différents modes [De Kleer et Williams, 1989]. Dans ce contexte, il est nécessaire d'identifier a priori les défauts ou dysfonctionnements du système pour en effectuer le diagnostic.

Le principe du diagnostic avec modèle de fautes

Dans cette approche la détection est réalisée par corroboration des observations avec les comportement calculés par au moins un des modèles de faute [Dubuisson, 2001]. Ceci est fait en comparant le comportement observé du système avec l'ensemble des comportements fautifs décrits dans le modèle. Dans ce cas, un conflit devient une assignation de modes de comportements qui sont en contradiction avec les observations (les prédictions du modèle sont incohérentes avec les observations). Dans cette approche, un diagnostic n'est plus un ensemble de diagnostics minimaux. Il s'agit d'une assignation de modes de comportements à toutes les composantes du système permettant de rétablir la cohérence avec les observations [De Kleer et Williams, 1989]. L'utilisation d'un modèle de fautes permet de chercher les causes qui expliquent les observations (symptômes).

Mais cette approche a aussi ses propres limitations car comme évoqué par [Dubuisson, 2001] :

"Le recensement préalable des défauts, des dysfonctionnements et de leurs relations éventuelles ne peut jamais être exhaustif et requiert en général une longue expérience dont la durée d'acquisition peut excéder le cycle de vie du système".

Alors il faut toujours considérer que, dans une approche à base de modèles de fautes, le modèle peut faire des prédictions incomplètes et l'ensemble des modes de fonctionnement des composants peut aussi être incomplet. C'est pourtant dans ce cadre que le diagnostic à base de modèles pour les systèmes à événements discrets a développé son résultat le plus connu appelé diagnostiqueur introduit dans [Sampath *et al.*, 1995].

Le modèle du système est le point crucial des approches de diagnostic à base de

modèles qu'il s'agisse des modèles de bon comportement ou des modèles de fautes. La section suivante est consacrée à ce point fondamental du DBM.

2.3 Le modèle du système

Nous commençons par donner plusieurs définitions du terme modèle pour remarquer que toutes convergent dans le fait que le modèle est seulement une approximation ou une vue partielle de la réalité (système dans ce cas) qu'il ne décrit pas exactement.

Définition de modèle

- " Un modèle est une description ou représentation conçue pour montrer la structure ou le fonctionnement d'un objet ou système [Landou et Besançon-Voda, 2001]."
- " Un modèle est une représentation formelle d'un système qui décrit la connaissance sélectionnée lors de la phase de modélisation [Dubuisson, 2001]."
- " Un modèle est une approximation de la réalité [Davis et Hamscher, 1988]. "

Le modèle utilisé dans la cadre du DBM est un modèle de connaissance qui se place dans le contexte d'une modélisation orientée composants. Dans une approche orientée composant, un modèle de comportement d'un système est obtenu à partir des modèles comportementaux de ses composants et de la structure du système. La structure du système est définie en terme de ports ou points d'échange pour interagir avec l'environnement [Darwiche, 1998]. Le comportement du système est donc défini en fonction des comportements de ses composants.

2.3.1 Construction du modèle du système

De façon générale, la construction du modèle peut être réalisée en trois étapes [Klein, 2005] [Olive, 2003].

1. **Modélisation des composants :** un composant est décrit dans le modèle par la représentation du comportement défini en terme de caractéristiques dynamiques (variables d'entrée/sortie, variables d'états, fonctions de transitions d'état).
2. **Connexion de modèles des composants :** la structure du système est utilisée pour coupler les différents modèles de comportement des composants. Le comportement global de l'ensemble des composants du système est un comportement unique qui ne peut pas être reproduit par un seul composant.
3. **Modélisation des entrées/sorties du système :** il s'agit d'intégrer au modèle les signaux observables du système, c'est-à-dire l'information pouvant être observée par l'intermédiaire des capteurs, dans le fonctionnement du système.

Lorsqu'il est impossible d'obtenir un modèle de comportement à partir des connaissances de base du système (lois de la physique par exemple), il est possible d'utiliser des techniques d'identification pour déterminer les paramètres essentiels du modèle. De telles techniques ne nécessitent pas de connaissance a priori du système à modéliser, mais s'appuient en général sur la connaissance de la structure du modèle. Il s'agit de collecter les données lors du fonctionnement du système pour construire le modèle de

comportement. Cette solution est appliquée quand les systèmes sont de grande taille et comportent un nombre important de composants en interaction.

Le domaine des systèmes à événements discrets dans lequel nous nous inscrivons a vu émerger des travaux portant sur cette question de l'identification à partir des observables [Meda *et al.*, 1998] [Meda *et al.*, 2005] et plus récemment [Klein *et al.*, 2005]. L'objectif dans ces travaux est d'obtenir un modèle interne du comportement du système, basé sur l'observation de ses entrées/sorties. Pour un modèle à événements discrets, les observations sont constituées par une séquence ordonnée de valeurs des entrées/sorties de ce système. Le modèle interne est une représentation mathématique du comportement du système. Les formalismes les plus utilisés sont les automates à états finis et les réseaux de Petri.

Dans [Klein, 2005] [Klein *et al.*, 2005], trois types de comportements sont distingués :

1. le comportement complet (réel) du système (généralement inconnu),
2. le comportement observé, c'est à dire l'ensemble des séquences observées sur les entrées/sorties,
3. le comportement identifié, constitué par un ensemble de transitions et d'états qui, par son évolution dynamique, est capable de générer les séquences d'entrées/sorties observées.

Quelques remarques s'imposent. La première est que le comportement observé est nécessairement un sous-ensemble du comportement complet sous les hypothèses de fonctionnement admises pour le système considéré. Par exemple, en diagnostic, il est très souvent fait l'hypothèse de la faute unique. Il ressort également de ces définitions que le comportement modélisé couvre le comportement observé. Il est donc tout à fait possible que certaines évolutions apparemment possibles dans le modèle identifié ne le soient pas dans le modèle complet. Enfin, il apparaît que plusieurs modèles différents peuvent constituer des représentations correctement identifiées d'un même flot d'observation.

Illustrons ceci à travers un exemple simple. L'observation est constituée par la séquence $< s_1, s_2 >$. Une identification correcte de ces observations est constituée par un automate à deux états $\{q_1, q_2\}$ tel que la transition $(q_1 \rightarrow q_2)$ génère l'observation $< s_1 >$ et que $(q_2 \rightarrow q_1)$ génère l'observation $< s_2 >$. Il est également tout à fait possible d'associer le réseau de Petri défini par :

$$P = \{p_1, p_2\}, T = \{t_1, t_2\}, Pre \left| \begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right|, Post \left| \begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right|, M_0 \left| \begin{array}{c} 1 \\ 1 \end{array} \right|,$$

Mais à l'évidence les deux modèles identifiés ne sont pas du tout équivalent en terme d'espace d'état.

La démarche suivie dans les travaux de [Klein, 2005] pour l'obtention du modèle est la suivante.

1. Pour chaque séquence observée, il est construit une séquence de n vecteurs codifiés (où n est un entier choisi arbitrairement). Chaque vecteur représente une lettre de l'alphabet. Les séquences de lettres sont ensuite transformées en séquences de mots.
2. Construction de l'automate qui accepte (reconnaît) le langage observé. Le premier mot de chaque séquence est associé avec un état initial, et le dernier mot est associé

avec un état final de l'automate. Ensuite, le principe de l'identification consiste à créer un état associé à chaque mot observé.

3. Renommer la fonction de sortie. Chaque état de l'automate construit correspond à une valeur unique d'entrée/sortie représentée par la dernière lettre du mot associé à cet état. L'état est alors renommé avec cette valeur.
4. Réduction de l'état final. Il s'agit d'associer les derniers états de chaque branche de l'automate qui sont identifiés par des lettres identiques.
5. Association d'états équivalents. Il s'agit de combiner les états équivalents. Deux états sont considérés équivalents s'ils sont associés aux mêmes sorties et s'ils ont le même ensemble d'états postérieurs.
6. Sous l'hypothèse que chaque séquence observée correspond à un fonctionnement cyclique particulier, alors les états initiaux et finaux de chaque séquence peuvent être fusionnés.

Cette démarche peut être résumée en trois étapes : observation des relations entrée/sortie, détermination des séquences (i. e. des cycles de fonctionnement) et finalement construction du modèle qui représente ces séquences. Dans [Meda *et al.*, 1998] [Meda *et al.*, 2005], le comportement identifié est modélisé par l'intermédiaire de réseaux de Petri interprétés.

Dans ces approches, le challenge est de minimiser les comportements excédentaires du modèle identifié pour qu'il corresponde aussi fidèlement que possible au modèle idéal représenté par celui du comportement réel. Ainsi, les conclusions formelles déduites du modèle identifié ont un écho dans la réalité du procédé modélisé.

La prochaine section est consacrée à la présentation des causes qui rendent difficile l'obtention d'un modèle fidèle du système.

2.3.2 Principales causes d'écart entre le modèle et le système

La phase de modélisation du système est soumise à de nombreux problèmes susceptibles de remettre en cause l'exactitude du modèle [Kamsu *et al.*, 2005].

- Le point de vue adopté par le concepteur peut être plus ou moins clair et évoluer tout au long du processus de modélisation.
- Les hypothèses retenues limitent l'expressivité et la précision du modèle qu'elles soient propres au système modélisé ou induites par le langage de modélisation utilisé (plus ou moins pertinent ou adapté à un domaine d'application).
- Les perspectives d'utilisation du modèle, en terme d'analyse par exemple, peuvent être difficiles à percevoir.
- La méconnaissance ou l'oubli de certaines caractéristiques importantes de l'environnement qui offrent une vision réduite des possibilités d'évolution du système.

L'absence de connaissance peut se justifier par les méconnaissances de situations complexes dans lesquelles le système peut évoluer, la connaissance sur un système complexe est elle-même un système complexe en soi [Le Moigne, 1990].

La qualité et la pertinence du modèle sont indispensables pour pouvoir ensuite juger ou tester une situation particulière dans le système modélisé. Un modèle non valide est plus qu'inutile, il peut être dangereux [Landou et Besançon-Voda, 2001].

D'après la définition du modèle il est constaté qu'il y a toujours une différence entre le modèle et le système car le modèle est tout simplement une approximation de la réalité telle qu'elle est perçue. C'est pour cette raison qu'il est nécessaire de considérer la possibilité d'écarts entre le système et le modèle (causés par des erreurs dans la phase de modélisation) dans la phase d'exploitation du système.

La suite présente un état de l'art du diagnostic à base de modèles dans le domaine des systèmes à événements discrets.

2.4 Approches de diagnostic à base de modèles dans les SED

Nous allons décrire de manière plus détaillée comment l'approche de DBM à été utilisée pour les SED. Même si cette approche a été implémentée dans le domaine des SED en utilisant deux types de modèles, modèle de bon comportement et modèle de fautes, c'est dans le second cas qu'il a pris le plus d'importance en donnant lieu à l'approche de diagnostic la plus connue : le diagnostiqueur.

Nous allons commencer par décrire les approches issues de l'utilisation d'un modèle de bon comportement pour continuer ensuite avec les applications sous des modèles de fautes.

2.4.1 Approches de diagnostic à base de modèle de bon comportement

Le principe de ces approches est de vérifier la cohérence entre les observations et le modèle du système surveillé qui décrit le comportement (normal). L'étape de détection consiste donc à comparer la séquence d'événements observée et la séquence d'événements produite par le modèle. Toute séquence observée qui n'est pas générée par le modèle est considérée comme une déviation anormale du comportement. Le but de cette étape est donc de détecter ces déviations. Une déviation du comportement constatée traduit une manifestation de défaillance dans le système (symptôme) ayant son origine dans une faute d'un composant. L'information sur les symptômes détectés est importante dans l'étape de diagnostic pour pointer sur les éléments du système qui sont la cause d'une déviation constatée (incohérence) entre les observations et le modèle.

Très peu d'approches (par rapport au nombre de celles qui se basent sur les modèles de fautes) ont suivi cette philosophie pour la détection et le diagnostic de défaillances.

2.4.1.1 Le diagnostic vu comme le rétablissement de la cohérence par la détermination des conditions opératoires non respectées

Ce principe a été appliqué [Holloway et Krogh, 1990] dans les systèmes manufacturiers où le modèle du système est utilisé pour calculer les comportements corrects attendus en termes de fenêtres temporelles pour l'occurrence des événements. Naturellement, les événements arrivant en dehors de leur fenêtre sont considérés comme des manifestations de comportements incorrects. Ces événements sont utilisés pour le raffinement des estimations des prochains comportements du système. Ici l'objectif du diagnostic est de déterminer l'ensemble des conditions nominales de fonctionnement provoquant, quand

elles sont violées, le comportement anormal observé. Cet ensemble de conditions non respectées constitue l'ensemble des hypothèses de diagnostic. Le nombre d'hypothèses peut être réduit en utilisant des observations du comportement normal du système exonérant certains composants ou validant des hypothèses produites dans le passé. La suite de ces travaux dans [Ashley et Holloway, 2001] utilise des séquences de conditions pour décrire le comportement du système au travers de réseaux de Petri où une condition est définie comme un signal des capteurs du système ayant une valeur vraie au fausse. Les réseaux de Petri sont également utilisés dans [Tabakow, 2007] où les comportements défaillants sont détectés lorsque le marquage du réseau ne respecte plus les propriétés structurelles des invariants de place. Le diagnostic est donné sous la forme d'un ensemble de places dont un marquage erroné peut expliquer le marquage observé et le non respect des invariants. Dans cette approche, en revanche, rien n'est dit sur la manière de reconstituer le marquage du réseau. Une solution pourrait être d'associer un ensemble de mesures à chaque place du réseau et de déduire le marquage observé de la valeur de ces mesures. Ceci permettrait de plus de raisonner avec un ensemble de mesures incomplètes. Des approches de diagnostic suivant cette logique ont été développées et intégrées dans une architecture de commande surveillance hiérarchisée [Combacau, 1991]. Le modèle est appelé modèle de référence, il est basé sur les réseaux de Petri à objet. Les mécanismes de détection implémentés sont du type chiens de garde pour surveiller les dates d'arrivée des compte rendus issus du procédé et du type de qualification du compte rendu pour vérifier si le compte rendu reçu correspond au compte rendu attendu [Combacau, 1991]. La suite de ces travaux est présentée dans [Chaillet-Subias, 1995] où il est mis en évidence le manque d'informations pour certaines fonctions de la surveillance, un centre d'information a été proposé et intégré à l'architecture pour donner des renseignements sur le procédé et pallier les limitations qui affectent tout principalement les fonctions diagnostic et reprise.

2.4.1.2 Le diagnostic vu comme rétablissement de la cohérence par modification des observations

Dernièrement des travaux dans le monde de l'automobile ont été développés dans [Soldani *et al.*, 2006] pour la détection de défaillances intermittentes dans les calculateurs embarqués. L'originalité de cette proposition réside dans une modification des séquences observées pour retrouver une séquence admissible dans le modèle du système. Cette démarche est liée à la recherche de fautes particulières, intermittentes et fugitives, se traduisant par l'occurrence spontanée d'un événement ou par la non production d'un événement prévu.

L'hypothèse implicite depuis le début de la technique DBM comme en témoignent les approches présentées est qu'une différence entre la réalité et la référence constitue un symptôme, c'est à dire, la partie visible d'un dysfonctionnement du procédé. Tout ce qui précède est tout à fait correct si nous considérons qu'un modèle est un reflet exact du système. Il est temps de relâcher l'hypothèse d'un modèle correct et de considérer qu'une divergence entre le comportement observé et le comportement décrit dans le modèle peut également être expliquée par une erreur dans le modèle servant dans la phase de détection et diagnostic. C'est précisément dans cette nouvelle hypothèse d'un modèle potentiellement entaché d'erreurs que notre travail s'inscrit.

Effectivement, ces erreurs dans le modèle dues aux aléas de modélisation peuvent

amener à ne pas représenter fidèlement la réalité du système, en obtenant ainsi un modèle qui ne décrit pas les propriétés, contraintes et exigences du fonctionnement imposées par le cahier de charges du système.

2.4.2 Approches de diagnostic à base de modèle de fautes

L'approche de diagnostic à base de modèles de fautes la plus connue est l'approche " diagnostiqueur " introduite dans [Sampath *et al.*, 1995]. Un diagnostiqueur est un automate obtenu par compilation d'un modèle du système surveillé contenant les événements de fautes. Cet automate n'est réceptif qu'aux événements observables définissant le comportement du système à diagnostiquer en présence de fautes modélisées par des événements inobservables. L'état du diagnostiqueur donne, à tout instant, une information sur les fautes pouvant expliquer le comportement observé. Dans cette approche, le diagnostic est un ensemble d'événements inobservables (fautes) expliquant les observations. La détection proprement dite d'une faute ainsi que son identification sont intégrées à la méthode. L'hypothèse de base de ces travaux consiste à affirmer qu'au bout d'un temps fini, les observations contiennent assez d'informations pour que la faute soit identifiée sans ambiguïté à partir des observations. Cette approche a pris une grande importance, elle est à la base de plusieurs travaux postérieurs où le diagnostiqueur est envisagé dans un cadre décentralisé ou distribué.

2.4.2.1 L'approche diagnostiqueur centralisé

Approche de l'Université du Michigan

Dans sa version originale [Sampath *et al.*, 1995], le diagnostiqueur est une entité centralisée chargée d'observer, de détecter et de diagnostiquer les défaillances du système global. C'est-à-dire, qu'à partir d'un modèle global du système surveillé et guidé par les observations reçues, le diagnostiqueur détermine l'occurrence des événements inobservables (défaillances) qui sont consistants avec les observations. Cette structure de diagnostic est montrée figure 2.1.

Le fonctionnement du diagnostiqueur est illustré dans la figure 2.2. Comme il est possible de l'observer dans cette figure, chaque état du diagnostiqueur contient un état estimé du système. Cet état n'est pas un état unique, il est composé de plusieurs états qui correspondent à différents modes de défaillances. Les étiquettes de ces états sont N pour décrire un comportement Normal et F_i pour décrire que le type de défaillance F_i a eu lieu. Donc, à partir de l'état initial, à la réception d'un événement σ_1 le diagnostiqueur estime que le système est amené à l'état étiqueté par $3F_1$, qui exprime la possibilité que le système se trouve dans l'état 3 avec l'occurrence d'une défaillance du type F_1, et par l'état $7N$, qui indique que le système peut être dans l'état 7 qui est un état Normal.

D'autres approches du diagnostiqueur centralisé peuvent être consultées dans [Chung *et al.*, 2003], [Ushio *et al.*, 1998].

Les limitations de cette version centralisée du diagnostiqueur, propres aux architectures centralisées (peu tolérantes aux défaillances, peu évolutives, etc.), ont donné lieu à des versions décentralisées de l'approche.

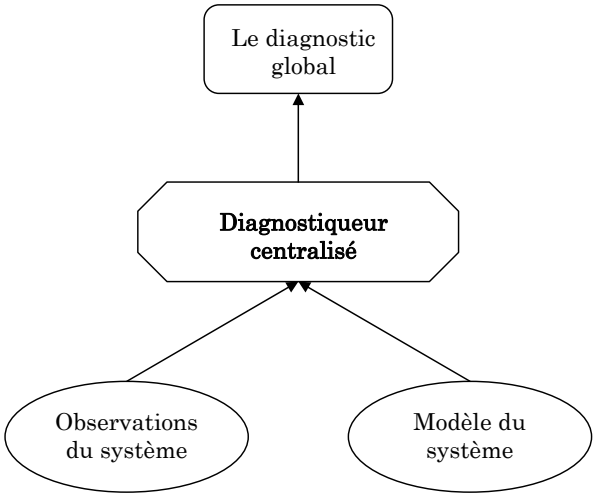


FIG. 2.1 – Diagnostiqueur centralisé

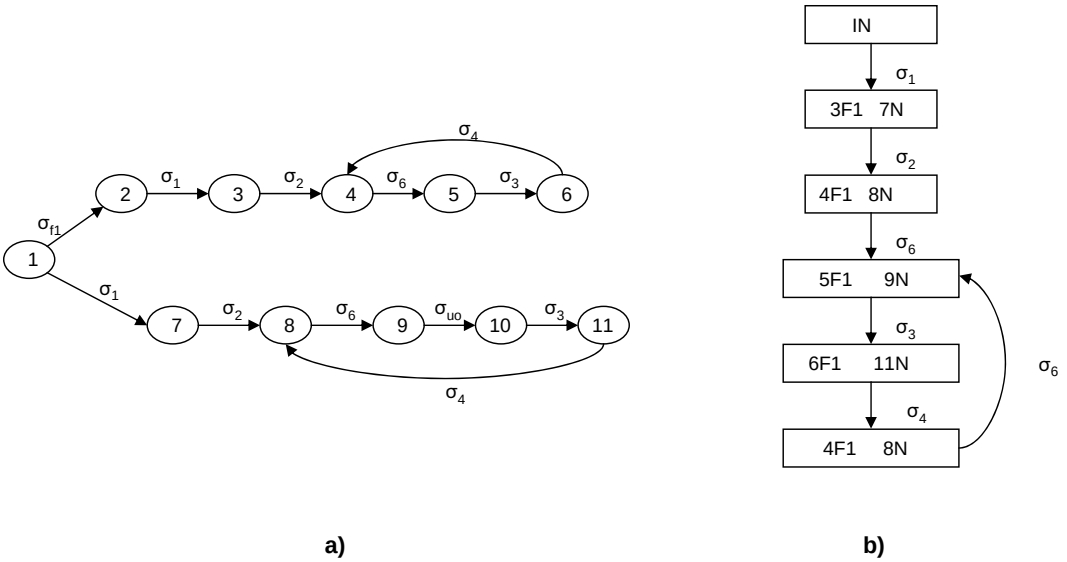


FIG. 2.2 – a) modèle du système b) le diagnostiqueur

2.4.2.2 Les approches décentralisées

L'approche de l'Université du Michigan

Dans le travaux de [DEBOUK *et al.*, 1998a], l'approche diagnostiqueur est utilisée dans une architecture décentralisée comme décrit dans la figure 2.3, constituée par deux sites locaux (diagnostiqueurs) qui communiquent avec un coordinateur. Un diagnostiqueur est composé par un module d'observation et un module de diagnostic. Chaque site utilise le modèle global du système comme référence pour le diagnostic. Ainsi, le site observe, traite ses observations et génère son résultat de diagnostic localement. Les résultats des diagnostics locaux sont transmis au coordinateur qui les fusionne pour aboutir à un diagnostic global du système. Si une faute est identifiée au niveau du module co-

ordinateur elle est transmise au module de récupération de défaillances. Comme dans [Sampath *et al.*, 1995] les diagnostiqueurs locaux fournissent des informations sur l'état du système et sur l'occurrence des défaillances. Une version étendue de cette approche et des protocoles de communication associés est proposée dans [Debouk *et al.*, 1998b].

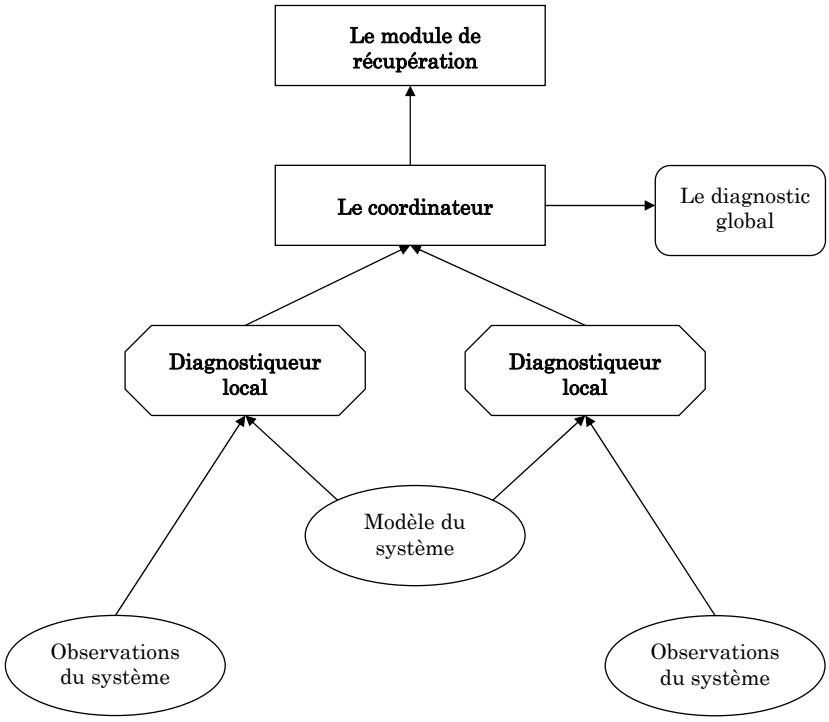


Fig. 2.3 – Approche décentralisée de l'Université du Michigan

Dans cette approche la mise en œuvre du diagnostiqueur est décentralisatisée mais le modèle du système utilisé pour générer les structures de diagnostic reste un modèle global. Nous avons évoqué précédemment la difficulté d'obtenir un tel modèle.

L'approche de l'IRISA

Cette approche [Pencolé, 2000] s'appuie sur un modèle décentralisé pour réaliser un diagnostic local en ligne. Le diagnostic consiste à inférer des informations sur l'état défaillant des composants du système et sur l'occurrence des événements de défaillance, à partir d'une séquence d'observations reçues (des alarmes). Un superviseur centralisé est chargé de recevoir toutes les observations en provenance du procédé. Ces observations sont ensuite envoyées au coordinateur qui les transmet aux diagnostiqueurs locaux accompagnées de l'information sur l'état du système. A partir de ces informations, chaque diagnostiqueur local détermine le sous ensemble de trajectoires (séquences de transitions et états) expliquant les observations locales. Ces diagnostics locaux sont renvoyés au coordinateur qui se charge d'appliquer une stratégie de combinaison appropriée pour élaborer un diagnostic global. La figure 2.4 montre cette approche du diagnostic.

Dans cette approche, le diagnostiqueur local a perdu la capacité d'observer. Il est au service des entités centralisées : le superviseur et le coordinateur. Le superviseur est en

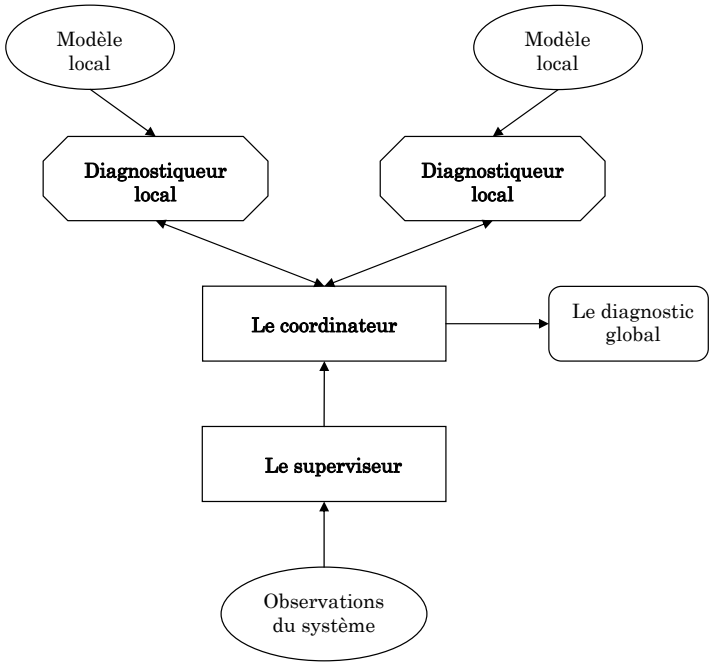


FIG. 2.4 – Approche décentralisée de l'IRISA

charge de récupérer toutes les observations issues du système. Le coordinateur déclenche les diagnostics locaux et fusionne leurs résultats.

2.4.2.3 Les approches distribuées

L'approche de l'Université de Brescia

Cette approche traite du diagnostic des systèmes actifs [BARONI *et al.*, 2000]. Dans cette approche, le diagnostic consiste en la reconstruction de la réaction du système à partir des observations et à l'identification des historiques de défaillances (les séquences de transitions d'un état initial à un état final). Le diagnostic est réalisé en trois étapes : la première constitue la formulation d'un plan de reconstruction du diagnostic à partir de diagnostics plus élémentaires. La deuxième est dédiée à la reconstruction de toutes les trajectoires possibles du système qui sont consistantes avec les observations. La troisième est la génération du diagnostic par isolation de toutes les trajectoires contenant une transition de faute. Dans cette approche les événements (messages) envoyés vers l'extérieur du système sont observés par un ensemble d'observateurs. Ces observateurs transmettent ces informations à plusieurs unités de traitement (diagnostiqueurs) qui reconstruisent le comportement d'un ensemble de composants (cluster) à partir d'un état initial jusqu'à l'état final. Ici la reconstruction du comportement du système est progressive (composant, cluster, système). La figure 2.5 montre l'architecture utilisée dans cette approche.

L'approche de l'Université de Toronto

[SU *et al.*, 2002] propose une méthode de diagnostic distribué à partir de diagnostiqueurs locaux. Chaque diagnostiqueur est associé à un composant et est élaboré à partir

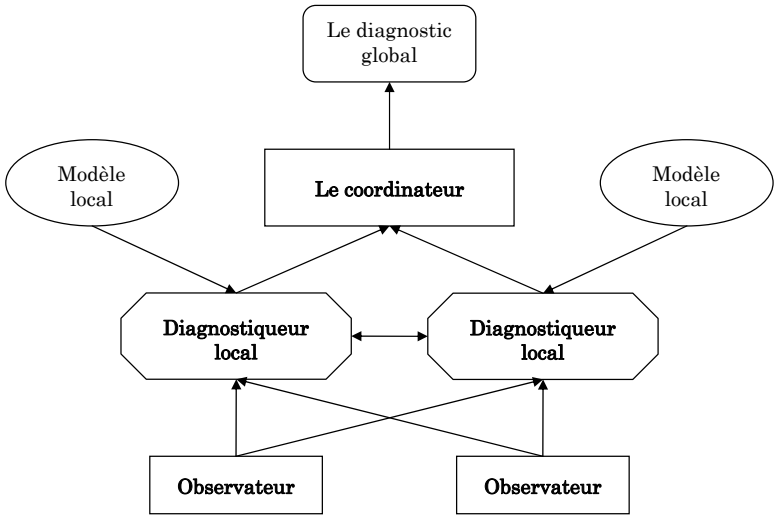


FIG. 2.5 – Approche distribuée de l'Université de Brescia

du comportement de ce composant et de ses interactions avec les autres composants. Les diagnostiqueurs communiquent directement entre eux selon les relations d'entrée/sortie entre les composants associés. Le diagnostic local détermine l'état normal ou fautif du composant d'après les observations locales. Les diagnostiqueurs communiquent entre eux pour partager leur information de diagnostic avec l'objectif d'améliorer leurs diagnostics locaux. Le diagnostic global est déterminé par recherche de la consistance entre les diagnostics locaux. La figure 2.6 montre l'architecture de cette approche.

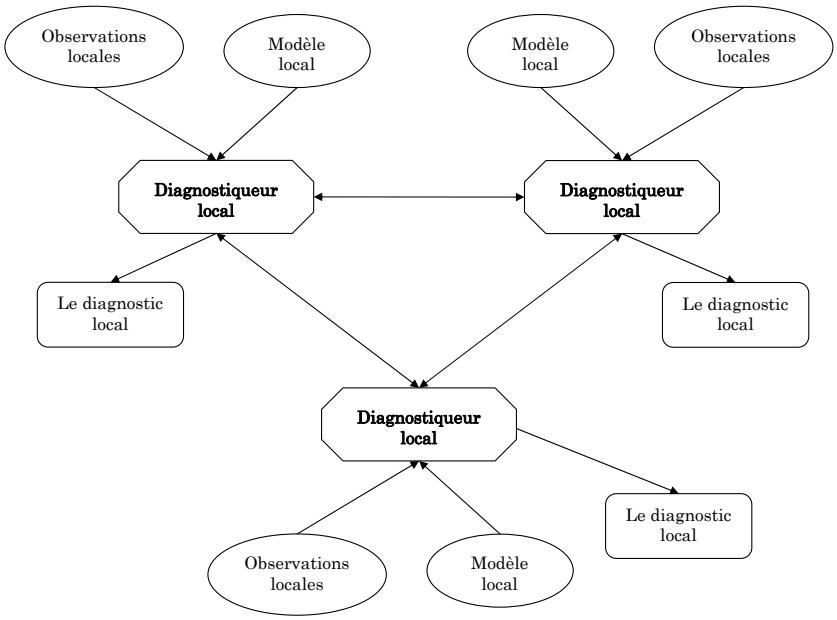


FIG. 2.6 – Approche distribuée de l'Université de Toronto

L'approche distribuée de l'Université de Michigan

Un algorithme de diagnostic distribué a été modélisé par réseaux de Petri étiquetés [Genc et Lafortune, 2003]. Cette architecture de diagnostic de la figure 2.7 est formée par deux diagnostiqueurs locaux, chacun d'eux surveillant une partie du système. Les diagnostiqueurs ont des places communes et communiquent entre eux après la réception d'une observation pour mettre à jour leur état. Dans cette approche les défaillances sont modélisées par des transitions inobservables, lesquelles sont étiquetées avec les défaillances qui affectent le système. Le diagnostic local consiste à déterminer d'une part l'état dans lequel se trouve le système après une observation et d'autre part l'occurrence et le type des défaillances. Le diagnostic global est obtenu en combinant les résultats des diagnostics locaux. Cette architecture distribuée est proche de la version décentralisée. La différence se situe au niveau des communications inter-diagnostiqueurs qui sont présentes seulement dans la version distribuée. Une architecture avec plus de deux diagnostiqueurs est proposée dans [Genc et Lafortune, 2005].

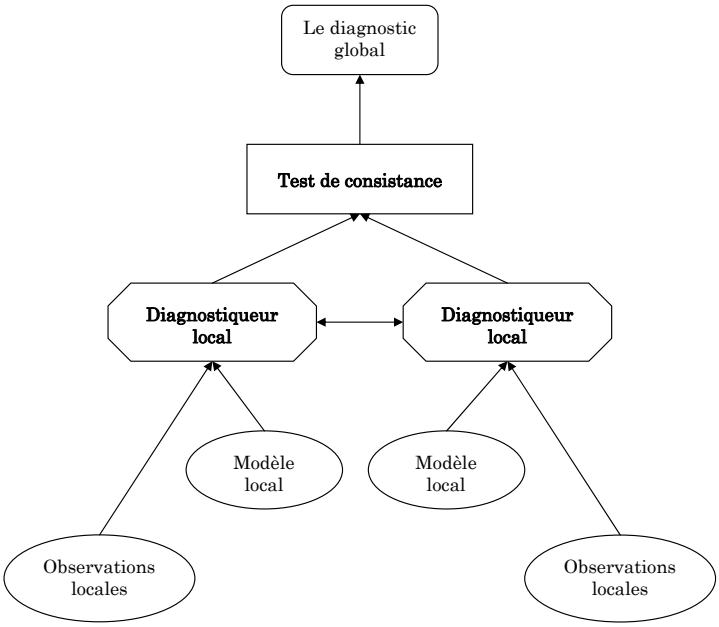


Fig. 2.7 – Approche distribuée de l'Université de Michigan

Le diagnostiqueur est un outil efficace pour la détection et l'isolation de défaillances dans les systèmes à événements discrets, de là, l'intérêt de l'adapter aux applications décentralisées et distribuées. Toutefois, l'approche diagnostiqueur présente certaines limites que nous détaillons ci-dessous.

– **Le modèle du système** utilisé dans l'approche diagnostiqueur comme dans d'autres approches [Lefebvre et Delherm, 2005] [Ghazel, 2005] [Jard *et al.*, 2005] décrit le comportement du système en présence de certaines défaillances connues. L'utilisation d'un tel modèle suppose l'exhaustivité dans l'anticipation des défaillances qui affectent le comportement du système. Autrement dit, cela demande la détermination préalable de toutes les défaillances possibles pouvant écarter le comportement du système de celui attendu. Un modèle

de fautes ne peut pas être exhaustif, car la taille du système et les façons dont les défaillances des composants peuvent se combiner font que l'hypothèse d'exhaustivité est peu réaliste. Alors, seules les fautes connues lors de la conception sont détectées et diagnostiquées. Il s'agit d'une limitation importante car l'efficacité de ces approches repose sur l'anticipation des fautes et de leur conséquences comme il a été mentionné dans [Dubuisson, 2001]. Il reste que l'approche diagnostiqueur fournit une solution peu coûteuse en temps de calcul en ligne permettant d'assurer une bonne couverture des fautes et des défaillances ayant été décrites lors d'une analyse préalable de sécurité comme l'AMDE. En ce sens, le diagnostiqueur est un bon candidat pour des solutions embarquées critiques nécessitant une certification comme dans le domaine de l'aéronautique ou de l'automobile.

– **Dépendance des diagnostiqueurs locaux :** dans les approches non centralisées, les diagnostiqueurs sont très dépendants des autres entités qui forment le système de surveillance. Leur tâche de diagnostic repose toujours sur les informations fournies par les entités centralisées comme le superviseur, le coordinateur ou sur d'autres entités locales de diagnostic.

Ces constatations sont à la base de notre travail dont les objectifs sont décrits dans le paragraphe suivant.

2.5 Contributions de la thèse dans le diagnostic de défaillance à base de modèles dans les systèmes à événements discrets

Depuis le début, la technique de Diagnostic à Base de Modèles a été appliquée dans différents domaines industriels (continus ou discrets), que ce soit en utilisant un modèle expliquant les fautes pouvant affecter le système à surveiller ou que ce soit en utilisant un modèle décrivant uniquement le comportement normal du système.

Dans le domaine discret cette technique a eu beaucoup plus de succès en utilisant des modèles intégrant les fautes du système.

Notre travail se situe dans le diagnostic des systèmes à événements discrets. Dans notre approche la détection des défaillances est réalisée en utilisant un modèle décrivant uniquement le comportement normal du procédé. Ainsi, nous assurons la détection de tout comportement qui n'est pas représenté dans le modèle de comportement normal. La connaissances a priori des fautes du système n'est pas requise pour pouvoir détecter les symptômes qui en résultent. Il faut toutefois bien préciser que si malencontreusement, une configuration du système est telle qu'une faute ne se traduit pas par un comportement déviant perceptible au niveau des observations, alors cette faute ne sera pas détectée. Cette hypothèse n'est pas réellement contraignante, car elle élimine simplement les cas pour lesquels une faute ne serait jamais détectable.

L'hypothèse implicite du modèle correct dans l'approche de diagnostic à base de modèles

La technique a toujours été appliquée en utilisant des hypothèses sur le modèle utilisé pour le diagnostic.

- Dans le cadre d'une approche à base de modèles de fautes, l'hypothèse consiste à supposer que le modèle représente les fautes et leurs conséquences d'une façon exhaustive.
- Dans le cadre d'une approche à base de modèles de comportement normal, cette hypothèse est implicite et consiste à supposer que le modèle utilisé est correct, c'est à dire qu'il représente fidèlement la réalité du terrain. Cette hypothèse sous laquelle tous les travaux présentés précédemment s'inscrivent est une hypothèse très forte.

Dans le cas des approches à base de modèles de comportement normal, affirmer qu'un modèle est correct revient à affirmer deux choses : que le modèle est complet et que le modèle est exempt de tout type d'erreurs de modélisation. Affirmer qu'un modèle est complet consiste à considérer que tous les comportements normaux du système sont connus et qu'ils sont modélisés. Cela revient enfin à considérer qu'il n'y a pas de méconnaissances dans le fonctionnement normal du système. La supposition du modèle exempt d'erreurs nous indique que le modèle montre de façon exacte le comportement du système, sans erreurs de logique, sans oubli des caractéristiques importantes du système, avec une parfaite maîtrise de la technique de modélisation garantissant que tout comportement du système est représenté sans erreur, etc.

Compte tenu de la difficulté d'obtenir un modèle correct et complet du système, l'hypothèse du modèle correct doit être reformulée. Cette reformulation consiste à accepter la possibilité d'erreurs dans le modèle qui représente le comportement du système.

En levant cette hypothèse, notre travail doit considérer la possibilité qu'il existe des erreurs dans le modèle du système pendant la phase de détection et que la détection d'une incohérence entre les observations et le comportement attendu issu de ce modèle ne traduit pas nécessairement un dysfonctionnement du procédé. En d'autres termes, ces erreurs dans le modèle peuvent générer des manifestations du système de détection qui ne sont pas a priori discernables de symptômes résultant d'une défaillance ayant une faute d'un composant comme origine.

Par exemple, supposons que nous disposons d'un modèle idéal représentant de façon correcte et complète, le comportement du système fonctionnant correctement (normal). Nous allons appeler ce modèle *modèle de comportement réel*.

Supposons aussi que nous avons un deuxième modèle, *le modèle de comportement normal* du système, obtenu avec les techniques traditionnelles de modélisation. Il représente le modèle utilisé par le système de diagnostic. Si nous réalisons une correspondance entre l'espace d'états de ces deux modèles, il est clair que ces espaces d'états ne sont pas identiques. Il existe des écarts entre ces espaces d'états, c'est à dire qu'il existe des états d'un modèle qui ne sont pas retrouvés dans l'autre modèle. Nous considérons que ces écarts, sont le résultat des erreurs contenues dans *le modèle de comportement normal*, engendrées dans la phase de modélisation, car comme nous l'avons déjà mentionné le *modèle de comportement réel* représente le modèle idéal. Graphiquement notre hypothèse d'erreurs dans ce *modèle de comportement normal* est représentée dans la figure 2.8.

Il existe une partie du modèle de *comportements réel du système* qui n'est pas incluse dans le *modèle de comportement normal du système*. Il existe aussi des comportements représentés dans le *modèle du comportement normal* qui n'appartiennent pas au *modèle de comportement réel*. La prise en compte d'une telle hypothèse change profondément la tâche de détection et de diagnostic de défaillances, car la détection des défaillances est basée sur les comportements décrits dans le *modèle de comportement normal du*

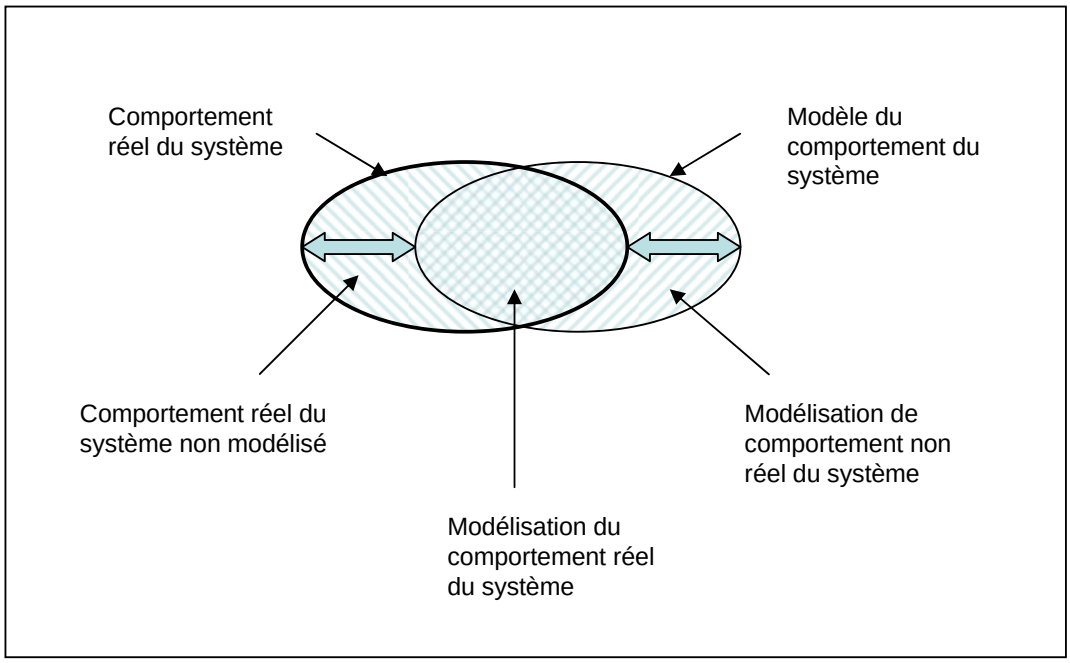


FIG. 2.8 – Représentation d'erreurs dans les modèles

système. En effet, comme nous l'avons déjà évoqué le système de détection ne met pas forcément en évidence un symptôme de défaillance, mais simplement une divergence entre les observations et le comportement nominal.

Supposons maintenant que le *modèle de comportement normal* contienne un comportement qui n'appartient pas au *modèle de comportement réel*. Lors de son observation, ce comportement sera considéré comme normal et il n'y aura pas détection de symptôme.

Nos travaux se distinguent ainsi des travaux développés au LAAS dans le cadre d'une architecture hiérarchique [COMBACAU, 1991] [CHAILLET-SUBIAS, 1995] [ZAMAÏ, 1997]. Dans ces travaux antérieurs, deux modèles sont considérés dans le module de commande/surveillance : le modèle du référence (modèle du procédé) et le modèle de commande. Les travaux de diagnostic développés dans ce cadre ont été réalisés sous l'hypothèse que la commande était exempte d'erreurs.

Nous considérons dans notre approche que le modèle de la commande n'est pas exempt d'erreurs. Dans notre approche, ce modèle et celui du procédé peuvent contenir des erreurs de modélisation.

Si les deux modèles comportent des erreurs, cela signifie qu'il existe des écarts entre l'espace d'états de ces deux modèles. L'intégration du *modèle de la commande* dans la figure 2.8 nous conduit au résultat de la figure 2.9.

Mais l'intégration de ce modèle n'est pas aussi évidente qu'il paraît car la question de la position relative de ce troisième modèle en regard des deux autres engendre une combinatoire importante. La figure 2.9 montre sous l'aspect de l'espace d'états, seulement deux cas possibles.

Ceci nous a conduit à proposer une technique permettant d'identifier la *configuration* qui peut exister entre ces trois modèles. Une configuration est définie comme la correspondance existant entre les ensembles de trajectoires des comportements décrits dans les modèles. La connaissance de la configuration est pertinente pour caractériser

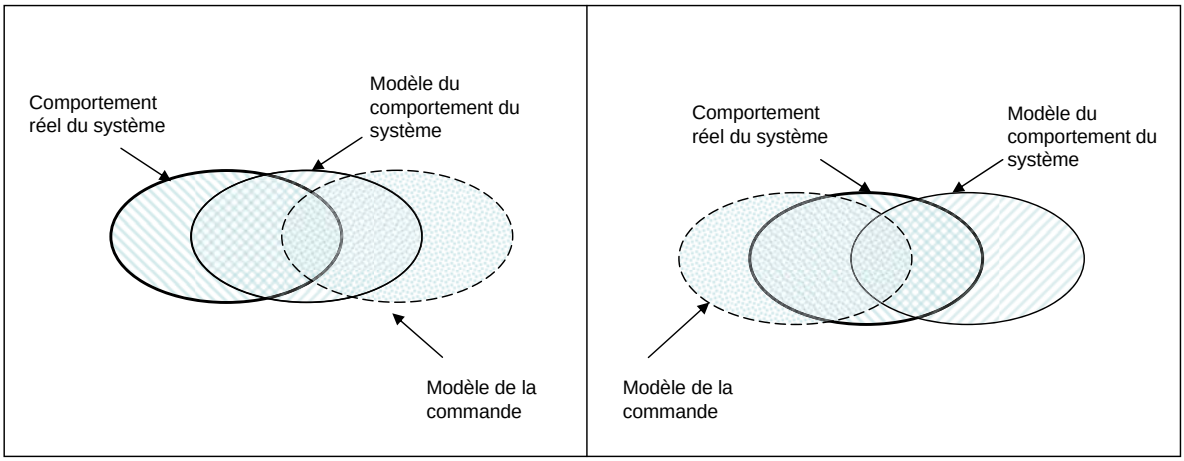


FIG. 2.9 – Intégration du modèle de la commande

la divergence entre les observations et les modèles i.e. pour distinguer si le symptôme détecté est la conséquence d'un dysfonctionnement du système ou si tout simplement il est causé par une erreur de modélisation.

Conclusions

L'avantage de l'approche du diagnostic à base de modèle de bon comportement est l'étape de détection, car il n'est pas nécessaire de connaître les dysfonctionnements du système pour pouvoir les détecter. Il suffit tout simplement d'être capable d'élaborer un modèle correct de ce système... Nous avons vu que cet aspect est le point faible de cette approche du DBM. Si le modèle utilisé est loin de décrire d'une façon convenable et valide la réalité qu'il suppose décrire, toutes les tâches pour lesquelles il a été conçu engendrerons des résultats aléatoires. Pour cette raison, il est indispensable de considérer que le modèle contient peut être des erreurs introduites volontairement ou non lors de la phase de modélisation. Ces erreurs causent des écarts entre le modèle et la réalité qu'il essaie de décrire. N'ayant pas de techniques capables d'identifier les parties du modèle qui sont correctes et celles qui ne le sont pas, il semble nécessaire de considérer qu'un écart entre les observations et le comportement de référence trouve sa cause dans une erreur de modélisation et non dans une faute d'un composant du système. Ces écarts peuvent se caractériser par des comportements normaux du système qui n'ont pas été répertoriés dans le modèle ou par la description de comportements antagonistes au comportement correct. Le même problème se pose au niveau d'un modèle de défaillances où il y aura toujours des comportements qui ne seront pas répertoriés ou de mauvaises assignations de modes de défaillances aux composants.

Comme nous l'avons déjà mentionné, ce type de problème dans le modèle avait été déjà évoqué dans les premiers travaux de DBM, bien que depuis le début, l'hypothèse d'un modèle correct ait été faite.

Le chapitre suivant est dédié à la description générale de notre approche de diagnostic à base de modèles pour les SED en considérant des erreurs dans les modèles.

Chapitre 3

Présentation générale de l'approche de détection et de diagnostic

Introduction

Ce chapitre est consacré à la présentation générale de notre approche de détection et diagnostic basés cohérence. Chaque étape de cette approche est développée en détail dans les chapitres suivants de ce manuscrit.

Il s'agit d'une approche à base de modèles décrivant uniquement le comportement normal. L'originalité dans notre approche réside dans l'hypothèse que les modèles du système ne sont pas nécessairement exempts d'erreurs. La conséquence de cette hypothèse est que, à la détection d'une incohérence entre observations et comportement attendu, il n'est pas possible de considérer qu'il s'agit d'un symptôme provoqué par un dysfonctionnement du système. Les erreurs du modèle peuvent également être à l'origine de cette perte de cohérence. Une telle situation peut apparaître, par exemple, lors de l'observation d'un comportement correct non décrit dans le modèle de bon comportement du système. La perte de cohérence est bien due dans ce cas à une erreur de modélisation et non à une défaillance du procédé.

3.1 Présentation du schéma général

Notre approche basée cohérence pour la détection et le diagnostic de défaillances pour les systèmes à événements discrets s'appuie sur plusieurs étapes :

1. la détection de la perte de cohérence entre les observations et le comportement attendu ;
2. l'identification de la configuration de l'ensemble de trajectoires des modèles. Une grande partie du travail présenté dans ce manuscrit traite de cette question. A notre connaissance, peu d'approches ont abordé sous l'angle des modèles la possibilité de la détection d'une anomalie en l'absence de défaillance du procédé (faute d'un composant entrainant une défaillance) ;
3. le diagnostic dont seul l'aspect restauration de la cohérence par modification des modèles est traité dans ce document.

La démarche générale de cette approche est illustrée dans la figure 3.1. Nous présentons tout d'abord la fonction détection de cette approche.

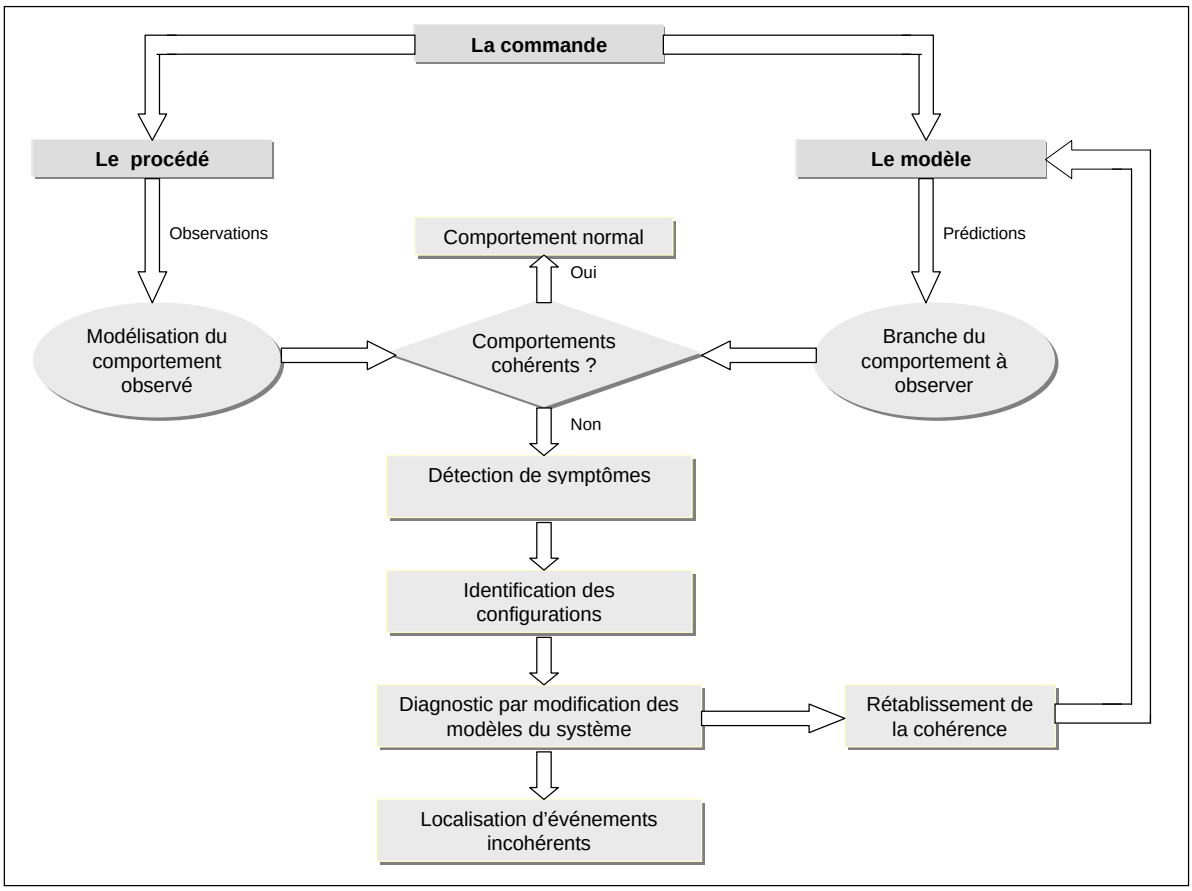


FIG. 3.1 – Schéma général de l'approche de détection et diagnostic basés cohérence

3.2 La fonction détection

Le principe de cette étape est de vérifier la cohérence entre les observations issues du fonctionnement du procédé et les modèles qui décrivent le comportement normal du système. Il s'agit donc de comparer la séquence d'événements observée traduisant la réalité, et la séquence générée par les modèles associés au comportement normal. C'est cette comparaison qui permet de réaliser la détection d'un symptôme de défaillance.

3.2.1 La rupture de la cohérence

Une observation issue du procédé qui ne correspond pas à un comportement dans le modèle rompt la cohérence entre la réalité et la référence, cette incohérence est généralement interprétée comme un dysfonctionnement du procédé. Toutefois, il y a trois causes possibles qui peuvent expliquer la rupture de la cohérence entre la réalité et les modèles de bon comportement.

- **Un changement des caractéristiques du procédé.** Ceci ce traduit par une réponse inhabituelle lors d'une sollicitation par le système de commande. Il s'agit donc d'une défaillance fonctionnelle au sens classique du terme : le procédé commandé ne se comporte pas comme l'indique sa spécification fonctionnelle. L'origine d'une défaillance est en général à rechercher comme une faute dans un composant, usure [Holloway et Krogh, 1990] une variation de l'environnement (par exemple, une intervention humaine intempestive sur le procédé) [Jakobson et Weissman, 1993] [Boubour et Jard, 1996] [Meida, 1997].
- **Un modèle faux.** Ce qui traduit des erreurs de modélisation lors de la conception dans le modèle supposé décrire le comportement normal du système et pris comme référence de ce bon comportement. Deux types d'erreurs de modélisation sont possibles : les erreurs dues à une description de comportements excédentaires, c'est-à-dire que le modèle englobe des comportements qui ne pas normaux dans la réalité et celles dues au contraire à un déficit de comportements autrement dit, il manque dans le modèle certains comportements normaux et possibles dans la réalité.
- **Des observations fausses.** Il s'agit soit d'une fausse lecture des capteurs, soit d'erreurs dans la transmission de l'information entre les capteurs et le système de surveillance. Cette lecture est interprétée par la surveillance comme la signature d'une évolution du procédé qui n'a pas eu lieu en réalité.

3.2.2 Les hypothèses dans notre approche

Dans notre approche nous faisons trois hypothèses, la première sur les observations, la deuxième sur les erreurs du modèle et la troisième sur les observations produites en présence de défaillances.

Hypothèse sur les observations : nous faisons l'hypothèse que les observations reçues ne sont pas entachées d'erreur. C'est-à-dire que ces observations représentent l'évolution réelle du procédé. Nous faisons cette hypothèse, même si dans [Sourrisse et Boudillon, 1997] est mis en évidence que 60% des cas de défaillances détectées sont liés à des problèmes de capteurs défaillants.

Hypothèse sur les erreurs du modèle : nous considérons que les modèles utilisés ne sont ni complets ni corrects, ce qui revient à considérer la possibilité d'erreurs dans les modèles.

Compte tenu de ces hypothèses, à l'apparition d'une incohérence, nous allons remettre en cause non seulement le fonctionnement du procédé surveillé mais également les modèles du système utilisés comme référence.

Hypothèse sur la trajectoire du procédé lors d'une défaillance : nous supposons que le procédé ne génère jamais, lors d'une défaillance, une séquence correspondant à une trajectoire comportant une erreur de modélisation.

3.2.3 Les modèles pour la détection

Pour mettre en évidence une rupture de cohérence nous nous appuyons non pas sur un modèle unique mais sur trois modèles :

- un modèle d'observations noté MOD_{OBS} construit en ligne, à la réception des observations issues du procédé. Ce modèle représente l'ensemble des trajectoires

d'états possibles que peut suivre le procédé suite à une séquence d'observations ;
- un modèle de comportement attendu noté MOD_{CA}, qui représente le comportement du procédé susceptible d'être observé en fonctionnement normal i.e le bon comportement ;
- le modèle de la commande, MOD_C qui décrit la façon selon laquelle le procédé doit être utilisé pour satisfaire l'ensemble de requêtes de commande imposées par l'utilisateur du système.

Ces deux derniers modèles sont mis en place hors-ligne à partir des données de conception et d'une connaissance experte du système surveillé, et sont utilisés tous les deux comme référence pour l'étape de détection. Dans la mesure où nous considérons possibles des erreurs au niveau des modèles, nous ne considérons pas la commande comme exempte d'erreurs.

Dans ce cas, la cohérence entre les observations et les modèles sera respectée si pour une commande en particulier le procédé génère une séquence d'observations qui correspond à une séquence d'événements observables décrite dans MOD_{CA} et que ces observations sont compatibles avec les trajectoires d'état de la commande représentées dans MOD_C. Au contraire, il y aura une rupture de cohérence dans les trois situations suivantes.
- L'observation reçue ne correspond ni au modèle de comportement attendu MOD_{CA}, ni au modèle de la commande MOD_C.
- L'observation reçue correspond au modèle de la commande MOD_C mais est incohérente avec le modèle de comportement attendu MOD_{CA}.
- L'observation reçue correspond au modèle de comportement attendu MOD_{CA} mais est incohérente vis-à-vis du modèle de la commande MOD_C.

3.3 Mise en évidence de la possibilité d'écart entre les modèles du système

Le fait qu'une observation soit cohérente avec un seul des modèles à la fois, peut être expliqué par les erreurs de modélisation au niveau de MOD_{CA} ou (dans le sens exclusif) MOD_C [Lopez Varela *et al.*, 2007]. Les erreurs de modélisation peuvent se manifester de deux manières : il peut s'agir d'un excédent d'information ou bien au contraire d'un déficit d'information. Quel que soit le type d'erreur, tous les deux se traduisent dans le modèle par un ensemble d'états qui n'ont pas de sens vis-à-vis du deuxième modèle. Autrement dit, les erreurs entraînent des écarts entre les espaces d'états et les comportements des deux modèles considérés. La figure 3.2 schématise trois des cinq configurations possibles donnant lieu à ces écarts entre MOD_{CA} et MOD_C. Cette figure exprime les cas suivants.
- Le cas a) peut être interprété comme une erreur au niveau du modèle de la commande MOD_C ayant pour effet de demander au procédé des comportements non prévus où qui n'ont pas de sens et qui par conséquent ne sont pas inclus dans MOD_{CA}. Cette configuration peut également traduire une erreur dans MOD_{CA} qui est incomplet et ne représente pas certains comportements du procédé bien que ceux-ci soient normaux.
- Le cas b) peut être dû à l'absence dans le modèle de la commande de certains com-

portements normaux du procédé. L'erreur peut également se situer dans MOD_{CA} qui représente alors des comportements impossibles en fonctionnement normal.
– Dans le cas c) les deux modèles peuvent présenter les deux types d'erreurs : des comportements excédentaires et en réalité anormaux, peuvent être modélisés dans MOD_{CA} mais MOD_C peut également être incomplet.

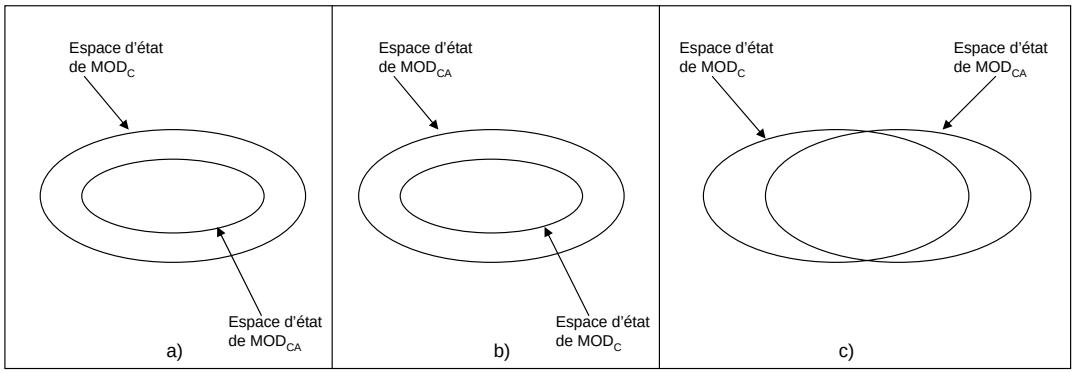


FIG. 3.2 – Représentation des configurations de l'ensemble des trajectoires de MOD_{CA} et MOD_C

Lorsqu'une incohérence est déterminée par MOD_{CA} et/ou MOD_C, ces deux modèles sont remis en cause car tous les deux sont potentiellement incorrects.

La technique pour déterminer une incohérence se base sur la recherche de comportements synchrones entre les modèles à partir de la mise en place des produits cartésiens [SCHNOEBEN, 1999] des modèles (MOD_{OBS} et MOD_{CA}) et (MOD_{OBS} et MOD_C). L'existence de tels comportements permettra de qualifier l'observation de cohérente. Si au contraire de tels comportements n'existent pas, l'observation sera qualifiée d'incohérente.

3.4 L'identification de la configuration de l'ensemble de trajectoires entre les modèles

Avant de considérer la phase du diagnostic après détection d'une incohérence, il est nécessaire de distinguer les incohérences causées par les erreurs de modélisation que nous appellerons *faux symptômes* et les incohérences causées par les dysfonctionnements du procédé appelées *symptômes de défaillance*. La réalisation de cette distinction soulève deux problèmes :

1. il est nécessaire de disposer de moyens nous permettant de discriminer les *faux symptômes* des *symptômes de défaillance.*,
2. il faut considérer que toutes les configurations pour les modèles du système sont possibles.

Pour résoudre ce premier problème nous utilisons un troisième modèle du système. Ce modèle est appelé modèle du comportement réel et est noté MOD_{CR}. Il s'agit d'un modèle représentant tous les comportements réels et normaux du système de façon complète et correcte. En utilisant ce troisième modèle nous pourrons distinguer les faux

symptômes des symptômes de défaillance. Avec MOD_{CR}, nous savons que toute observation incohérente provoquée par une erreur dans MOD_{CA} et/ou MOD_C sera cohérente avec MOD_{CR} et toute observation incohérente avec MOD_{CA} et/ou MOD_C provoquée par un dysfonctionnement du procédé sera aussi incohérente avec MOD_{CR}.

Cependant, la prise en cause du deuxième problème mentionné complique la distinction des symptômes. En effet, le troisième modèle MOD_{CR} doit être intégré à toutes les situations d'écart possibles entre MOD_C et MOD_{CA}.

L'étape d'identification, qui est une étape importante de notre approche (voir figure 3.1), s'appuie sur la mise en place d'un graphe construit à partir d'informations émanant des observations. Comme nous le verrons plus en détail dans la suite de ce document, ce graphe ayant sept niveaux de profondeur, un maximum de sept observations apportant des informations différentes est nécessaire pour identifier la configuration des modèles correspondant aux observations. Une fois obtenu cet ensemble de configurations, la distinction entre les faux symptômes et les symptômes de défaillances peut être réalisée. Naturellement, tant que les sept informations complémentaires ne sont pas collectées, une incertitude sur la configuration des modèles reste et les conclusions sont entâchées d'incertitudes.

3.5 Le diagnostic

L'étape de diagnostic dans notre approche est dédiée à rétablir la cohérence entre la réalité et la référence.

Etant donné l'hypothèse des observations correctes et celle d'erreurs possibles dans les modèles, pour rétablir la cohérence entre la réalité et la référence, il est nécessaire de modifier la référence, c'est-à-dire les modèles du système MOD_{CA} et MOD_C. L'objectif de ces modifications est que ces modèles puissent représenter à nouveau le comportement observé tel qu'il est dans le modèle d'observations MOD_{OBS}.

Pour cette étape il est important de pouvoir distinguer les incohérences causées par des erreurs de modélisation de celles dues à des symptômes de défaillance. Cette connaissance est en effet utilisée pour caractériser les modifications réalisées sur les modèles.

L'étape de diagnostic va donc être une étape de diagnostic par modification des modèles du système comme indiqué figure 3.1. Comme nous l'avons mentionné dans le premier chapitre, les modèles que nous considérons sont mis en place par réseaux de Petri. Nous utiliserons pour le raisonnement la représentation matricielle. L'objectif du diagnostic est donc de trouver un ensemble de matrices d'incidence modifiées, appelées matrices de Rétablissement de Cohérence (notées C_{RC}), rétablissant la cohérence en incluant le comportement observé non représenté dans les modèles initiaux.

Plusieurs types de modification à apporter aux modèles sont envisageables à condition que ces modifications respectent les contraintes suivantes :

1. les nouvelles matrices trouvées doivent respecter la représentation des comportements observés du procédé qui ont été trouvés cohérents auparavant ;
2. les modifications réalisées doivent garantir certaines propriétés du modèle réseau de Petri du système. Ces propriétés sont déterminées à partir d'une connaissance (minimale) du système surveillé. Par exemple les propriétés inscrites dans les invariants linéaires de place ainsi que les bonnes propriétés d'un système représentées

dans le modèle initial comme la bornitude, la réinitiabilité et la vivacité doivent être conservées.

Dans ces conditions, les modifications que nous pouvons effectuer sur les réseaux de Petri utilisés dans notre approche sont les suivantes.

a) Modifications des poids des arcs.

Il s'agit de modifier les effets des franchissements de transitions sur le marquage, i.e les coefficients de la matrice d'incidence C, sans modifier la dimension de la matrice (nombre de lignes et de colonnes), c'est-à-dire, sans changer les ensembles P et T du modèle. Plus spécifiquement, cette modification est réalisée dans la matrice d'incidence Pre car au niveau du modèle réseau de Petri, une incohérence se traduit par le non franchissement de la transition associée à l'événement observé, cette transition n'étant pas sensibilisée. Le franchissement de cette transition nécessite de modifier le poids des arcs la reliant à ses places d'entrée, voire de la relier à des places marquées dans le marquage courant du réseau.

b) Modifications des ensembles P et T.

Il s'agit de modifier le nombre de lignes et de colonnes de la matrice d'incidence en les augmentant ou en les diminuant, sans changer les coefficients de la matrice associés aux éléments du réseaux qui ne sont pas modifiés. En fait, par cette modification il s'agit d'intégrer dans le réseau de nouvelles places et de nouvelles transitions de telle sorte que le modèle puisse représenter le comportement observé.

c) Modification de l'état du réseau.

Par cette modification, il s'agit de modifier le marquage du réseau de Petri, pour que le nouveau marquage sensibilise la transition associée à l'événement observé.

Les modifications induisent en général de nouvelles propriétés qui n'étaient pas présentes dans le modèle initial du système. Par ailleurs, le nouvel espace d'état du réseau doit inclure les comportements considérés cohérents avec les observations reçues avant détection de l'incohérence ainsi que le comportement considéré comme incohérent jusqu'à la modification du modèle. Le résultat de cette modification est l'obtention d'un modèle des comportements du système incluant les comportements décrits par le *modèle de fonctionnement normal* initial.

La méthode de diagnostic présentée dans ce document au niveau du chapitre 6 ne traite que du rétablissement de la cohérence par la modification des coefficients de la matrice d'incidence Pre. Les autres types de modifications ne sont pas développés dans cette thèse.

Conclusions

Nous avons présenté dans ce chapitre le schéma général de notre approche de détection et diagnostic de défaillances, qui repose sur trois étapes. La première étape est naturellement la détection d'une rupture de cohérence entre la réalité capturée dans le modèle d'observations MOD_{OBS} et la référence du bon comportement du système représentée par les modèles du comportement attendu (MOD_{CA}) et de la commande (MOD_C). Nous avons mis en évidence que des erreurs au niveau des modèles provoquent de faux symptômes et en même temps des écarts entre les espaces d'états des modèles.

La deuxième étape de notre approche a donc pour but de déterminer l'existence de telles erreurs et d'identifier la situation d'écart associée. Nous proposons pour cela, d'utiliser le modèle du comportement réel du système (MOD_{CR}) et nous avons proposé une méthode permettant d'identifier la configuration entre les modèles en tirant profit des informations obtenues à partir des observations et en mettant en place un graphe d'identification des configurations.

La dernière étape de cette approche est celle du diagnostic. Ayant détecté un symptôme de défaillance par la rupture de cohérence entre la réalité et la référence, le diagnostic consiste à rétablir cette cohérence. Compte tenu des hypothèses de base de cette approche (modèles non exempts d'erreurs et observations saines), le rétablissement de la cohérence est réalisé via la modification des modèles du système.

Le prochain chapitre décrit de façon détaillée la première étape de l'approche proposée qui est l'étape de détection.

Chapitre 4

La fonction Détection

Introduction

Le système réagit à des événements en réalisant une évolution d'état et parfois en émettant des événements qui à leur tour provoquent une réaction du système. Un événement peut être associé à une action spécifique provoquée (actions de commande), à une occurrence spontanée (panne d'une machine) dictée par la nature ou il peut résulter d'un ensemble de conditions.

Ce chapitre est consacré à la détection des événements qui font évoluer le système de façon inattendue (e.g. défaillances), en écartant le système de son comportement normal. La technique que nous utilisons pour la détection de ces comportements inattendus est à base de modèles décrivant uniquement le comportement normal du système. Le principe de cette technique consiste à confronter le comportement observé du système avec le comportement capturé dans le modèle qui représente le fonctionnement correct du système. Toute incohérence entre l'observation et le modèle donnera lieu au lancement d'un processus de diagnostic visant à replacer le système et ses modèles en parfaite cohérence.

Nous commençons le chapitre par quelques définitions importantes pour la compréhension de notre approche. Ensuite, nous donnons la description des modèles qui représentent le bon comportement du système. Deux modèles sont utilisés comme référence de comportement normal : le modèle de la commande et le modèle de comportement attendu. Après la description de ces deux modèles, nous considérons un troisième modèle, obtenu à partir du comportement observé et construit en ligne à la réception de chaque nouvelle observation. Ce modèle est appelé modèle d'observations et représente le comportement réel du procédé. Une fois ces différents modèles en place, nous présentons notre approche de détection par vérification de cohérence entre les modèles utilisés comme référence de bon comportement et le modèle d'observations décrivant le comportement réel.

4.1 Définitions

Nous donnons maintenant quelques définitions qui ont pour objectif de mieux fixer le cadre de notre travail et qui sont nécessaires à la compréhension de notre approche.

Evénement. Un événement e est un phénomène sans durée qui cause un changement d'état dans un modèle de Système à Evénements Discrets. Le SED est dit réceptif aux événements qui provoquent son évolution.

Nous appelons E l'ensemble des événements auquel le modèle est réceptif. Cet ensemble E peut être partitionné en l'ensemble d'événements observables noté E_{obs} et l'ensemble d'événements non observables appelé E_{nobs}.

$$E = E_{obs} \cup E_{nobs}$$

Evénement observable. Un événement e est dit observable s'il est en permanence possible d'associer son occurrence à une observation issue des capteurs équipant le système modélisé.

Evénement non observable. Un événement e est dit non observable, s'il n'est pas possible de constater en permanence son occurrence par l'intermédiaire des capteurs équipant le système modélisé.

Dans ces travaux nous nous focalisons uniquement sur les événements observables du système pour développer notre approche, étant donné que nos modèles du système représentent uniquement son comportement normal et observable. Par la suite et par souci de simplicité, nous utiliserons donc le terme événement pour faire référence à un événement observable.

Définition d'observation. Une observation o est définie comme la réception d'un signal émis par les capteurs du système indiquant l'occurrence d'un événement observable. En général, l'événement a pour origine le changement d'une valeur mesurable par les capteurs.

D'après cette définition, chaque observation o est associée à un événement observable e. La fonction Eo associe un événement observable à chaque observation :

$$OBS \overset{Eo}{\longmapsto} E_{obs}$$

$$o \overset{Eo}{\longmapsto} Eo(o) = e$$

où OBS est défini comme l'ensemble des observations.

Dans les modèles sur lesquels nos travaux s'appuient, les événements sont associés aux transitions du modèle par la fonction E_v

$$T \overset{Ev}{\longmapsto} E$$

$$t_i \overset{Ev}{\longmapsto} Ev(t_i) = e_i$$

Comportement du système : le comportement d'un système est défini par l'ensemble de trajectoires (états et transitions) constituant l'ensemble des évolutions possibles entre les états de l'espace d'état du système et les événements provoquant ces évolutions. Une trajectoire T_R est formée par un ensemble d'états $\{M_1, M_2, \ldots, M_n\}$ et des événements $\{e_1, e_2, \ldots, e_{n-1}\}$ provoquant les transitions entre ces états :

$$T_R = (M_1, e_1, M_2, e_2, \ldots, e_{n-1}, M_n)$$

4.2 Les Modèles du système

Comme nous l'avons présenté dans le chapitre 1 les systèmes de production font partie des systèmes auxquels nous nous intéressons. Un système de production est conçu pour fabriquer un ensemble de produits. La fabrication de ces produits nécessite une suite de transformations élémentaires appelées opérations [Giard, 2003]. Chaque opération impliquée dans la transformation d'une pièce peut être exécutée par une ou plusieurs machines à l'aide des différents outils de production. Pour enchaîner certaines opérations, le produit doit être acheminé d'un poste de travail à l'autre au sein du système. Ces transferts sont assurés par des moyens de transport variés (tapis, chariots radio-guidés, etc.). Dans notre travail, chaque opération participant à la transformation ou à l'acheminement d'un produit constitue une activité.

4.2.1 Choix d'un outil de modélisation

Les activités d'un système de production peuvent être modélisées par différents outils utilisés pour représenter un SED (automates, statecharts, RdP). Nous choisissons les réseaux de Petri pour leur capacité à représenter les caractéristiques propres d'un système de production, comme les activités concurrentes, la synchronisation, le partage de ressources, le stock de produits, des activités en exclusion mutuelle, etc. Un autre avantage des réseaux de Petri, est leur capacité à représenter de manière compacte l'ensemble des états dans lesquels peut se trouver le système via la distribution des jetons. La possibilité de vérifier formellement grâce au formalisme mathématique des RdP, certaines propriétés du modèle (vivacité, réinitiabilité, bornitude, invariants etc...) est également un atout de cet outil exploitable notamment lors de l'étape de diagnostic par modification du modèle.

4.2.2 Modélisation d'une activité

Sous le formalisme des réseaux de Petri, une activité peut être modélisée selon le principe de la figure 4.1 [Combacau, 1991] [Chaillet-Subias, 1995]. Supposons que l'activité modélisée dans cette figure soit une activité de `transport d'une pièce` réalisée par un robot. Dans cette figure les places p_1 et p_2 représentent les conditions nécessaires qui doivent être satisfaites pour l'exécution de l'activité `transport d'une pièce`. Cet ensemble de places est appelé préconditions de l'activité. Ces préconditions peuvent être interprétées comme `robot libre` et `pièce disponible` respectivement.

La place p_3 est réservée pour indiquer que l'activité `transport d'une pièce` est en cours d'exécution, c'est-à-dire, que le robot est en train d'effectuer le déplacement de la pièce.

Les places p_4 et p_5 indiquent l'état du système après la fin de l'activité. Cet ensemble de places est appelé postconditions. Dans notre exemple, ces places peuvent représenter le fait que le robot est à nouveau libre et que la pièce est dans le poste de travail requis. Les transitions dans ce réseau indiquent le début de l'activité de `transport d'une pièce` (transition t_1) et la fin de cette activité (transition t_2). Le marquage i.e. la répartition des jetons dans les places, donne la dynamique du système modélisé. Cette activité `transport d'une pièce` peut être affinée. Il est possible par exemple de la décomposer en trois activités, `prendre pièce`, `mouvement du robot` et `poser pièce`. Le niveau de finesse de description d'une activité dépend du but de la modélisation.

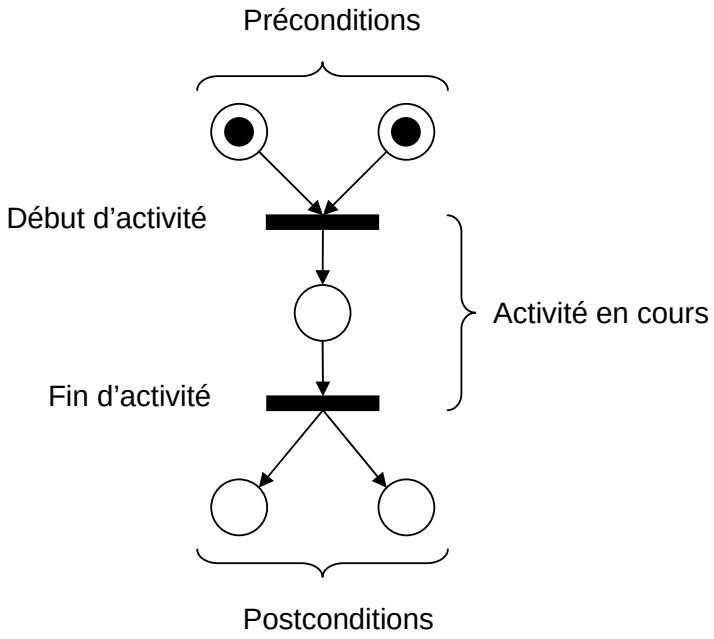


FIG. 4.1 – Modélisation d'une activité

L'enchaînement entre les différentes activités du procédé donne le modèle complet du système qui résulte de la composition des activités selon quatre opérations de base :

- Activités séquentielles : A_1 suivie de A_2, est représenté par la fusion d'une post-condition de A_1 avec une précondition de A_2 ;
- Activités concurrentes : A_1 et A_2 ne partagent aucune précondition (extension du concept de parallélisme en réseau de Petri) ;
- Activités en exclusion mutuelle : A_1 et A_2 partagent une précondition booléenne qui assure l'exclusion
- Synchronisation d'activités : A_1 et A_2 sont lancées simultanément, cela revient à fusionner leurs transitions `début`.

Nous avons déjà signalé que le niveau de détail d'une activité dépend fortement du niveau d'abstraction requis par l'application dans laquelle elle apparaît. Dans notre cas, le niveau de détail correspond au niveau d'équipement tel qu'il est défini dans [HUANG, 1996].

Nous allons nous appuyer sur cette notion d'activité pour la mise en place du modèle du comportement attendu et du modèle de la commande. Ces modèles sont détaillés dans la suite de ce chapitre.

4.2.3 Les modèles construits hors ligne

Deux modèles du système sont mis en place hors ligne : le modèle de comportement attendu et le modèle de la commande. Ces deux modèles sont mis en place à partir de la connaissance experte du système surveillé et sont utilisés tous les deux comme référence du fonctionnement correct du système dans l'étape de détection.

4.2.3.1 Le modèle de comportement attendu

Avant de décrire ce modèle nous allons commencer par décrire le modèle du procédé qui se trouve à la base du modèle du comportement attendu. Ce modèle a été présenté dans les travaux de [Combacau, 1991] [Chaillet-Subias, 1995] [Zamaï *et al.*, 1998], il a été utilisé comme référence du comportement normal du procédé. Le modèle du procédé noté MOD_{PR}, décrit toutes les activités que le procédé peut exécuter en fonctionnement normal. Il résulte de l'étude du cahier des charges du système et regroupe toutes les contraintes nécessaires pour respecter les lois de fonctionnement. La notion d'activité est utilisée pour modéliser le comportement du procédé dans MOD_{PR}.

Le modèle du comportement attendu, noté MOD_{CA}, représente le comportement du procédé susceptible d'être observé en fonctionnement normal. Ce modèle est implémenté par RdP ordinaires et ne décrit qu'un sous ensemble des comportements du procédé capturés dans MOD_{PR}.

En effet, MOD_{CA} décrit uniquement le comportement observable. Le comportement observable est défini par les activités dont l'exécution peut être constatée par les observations issues des capteurs du système. Ainsi, MOD_{CA} peut être vue comme l'abstraction du comportement observable dans le modèle du procédé. Formellement, cela se traduit par un modèle constitué d'un sous ensemble T_{CA} de l'ensemble des transitions T_{PR} de MOD_{PR}

$$T_{CA} = \{t_i | t_i \in T_{PR} \wedge E_v(t_i) \in E_{obs}\}$$

Dans MOD_{CA} les places entre une transition de début d'une activité et une transition de fin de l'activité disparaissent à la fusion de ces deux transitions

Cette abstraction est illustrée dans la figure 4.2. Dans cette figure, le RdP du MOD_{PR}, montre deux types de transitions, les transitions observables et les transitions non observables. Les transitions foncées correspondent aux transitions observables et les transitions non observables sont représentées en clair. Dans ces travaux les transitions non observables ne font pas références aux fautes. Nous appelons ces transitions inobservables car l'effet sur le procédé est inobservable. Une transition non observable correspond au début d'une activité alors qu'une transition observable est associée à la fin d'une activité qui sera constatée par la réception d'un compte rendu d'activité i.e des évolutions observable du procédé.

Une fois établi le modèle de comportement attendu MOD_{CA}, nous pouvons considérer le modèle de la commande.

4.2.3.2 Le Modèle de la Commande

Le Modèle de la Commande, identifié par MOD_C, décrit la façon selon laquelle le procédé doit être utilisé pour satisfaire l'ensemble de requêtes de commande imposées par l'utilisateur du système. Ce modèle indique par exemple pour une requête donnée, les activités à réaliser et leur séquencement, les différentes ressources nécessaires (machines, outils,..) etc. La commande peut aussi fixer le temps alloué à l'exécution de ces activités.

Les comportements décrits par le modèle de la commande MOD_C doivent normalement être limités aux comportements décrits par le modèle du procédé MOD_{PR}. En effet, ce dernier contient tous les comportements potentiels du procédé garantissant les propriétés du système et met en jeu tous les états utilisables du procédé alors que la commande s'appuie sur les états autorisés [Chaillet-Subias *et al.*, 1997].

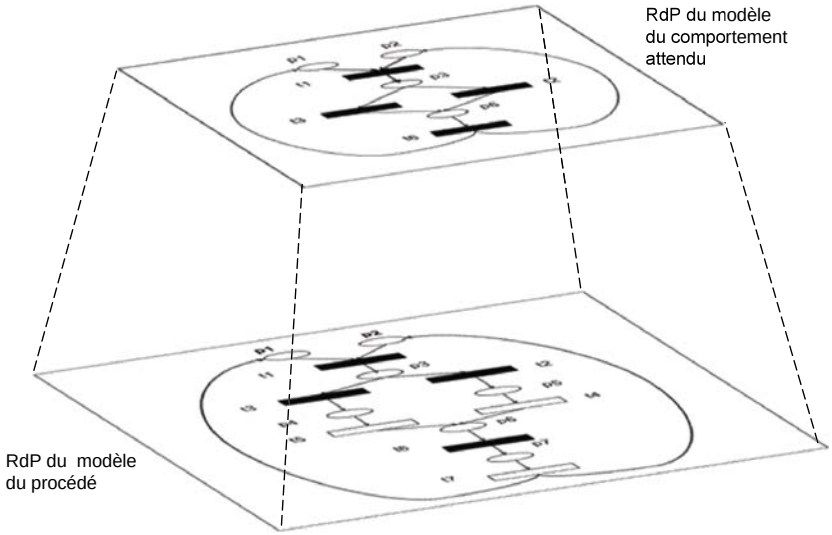


FIG. 4.2 – Abstraction du comportement observable

Néanmoins, peut être parce que ces deux modèles n'ont pas été conçus au même moment, ou par la même personne, ou tout simplement en raison d'une simple erreur d'écriture au moment de faire les correspondances entre les deux modèles. Bien encore pour d'autres raisons évoquées dans le chapitre 2, il est tout à fait possible que le modèle de la commande ne se limite pas strictement aux comportements décrits dans MOD_{PR}, et considère des comportements mettant en danger l'intégrité des moyens de production.

Ce modèle de la commande est aussi implémenté par un réseau de Petri ordinaire.

L'étape de détection basée cohérence que nous avons développée s'appuie sur un troisième modèle qui lui, va être construit en ligne. Ce troisième modèle appelé modèle d'observations est décrit dans la partie suivante.

4.3 Le modèle d'observations

Rappelons que l'une des hypothèses de notre travail est que *les observations reçues sont correctes.*

Cette hypothèse nous indique implicitement qu'à la réception d'une observation, cette observation traduit le comportement réel du procédé. Dans ces conditions :

- dès qu'une observation est reçue nous considérons que l'événement observable a eu lieu. Cette hypothèse élimine les observations fausses provoquées par des erreurs dans l'observation (problèmes de capteurs, problème de communication, etc), comme nous l'avons déjà indiqué dans le chapitre précédent ;
- l'ordre de réception des observations renseigne sur l'ordre dans lequel les événements observables ont eu lieu au niveau du procédé.

Supposons par exemple que le système de surveillance a reçu la séquence d'observations

$$S_{obs} = (o_1, o_2, \ldots, o_j)$$

Cette séquence d'observations va être modélisée par la séquence d'événements

$$S_{ev} = (E_o(o_1), E_o(o_2), \ldots, E_o(o_j))$$

Cette séquence d'événements constitue la partie observable de la trajectoire caractérisant le comportement réel du procédé et va nous servir de point de départ pour la recherche des erreurs dans les modèles du système (MOD_{CA} et MOD_C).

Le principe est le suivant : si pour le modèle MOD_X dans le marquage courant M_c, il existe au moins une trajectoire

$$T_R(M_c) = (M_c, E_o(o_1), M_1, E_o(o_2), \ldots, E_o(o_j))$$

alors nous considérons que le modèle du système représente de façon correcte le comportement observé puisque l'observation est cohérente avec le modèle de bon fonctionnement MOD_X.

Nous en déduisons que le modèle MOD_X est considéré correct tant qu'il peut accepter les comportements observés. Dans le cas contraire, puisque nous faisons l'hypothèse que les observations ne sont pas entachées d'erreur, nous serons amenés à conclure sur la présence d'erreurs dans le modèle MOD_X.

Après ces explications nécessaires relatives à la notion d'observation, nous donnons dans la partie suivante la définition de ce modèle d'observations et le détail de sa construction.

4.3.1 Définition du modèle d'observations

Le modèle d'observations noté MOD_{OBS}, est construit en ligne, à la réception des observations issues du fonctionnement du procédé, pour représenter le comportement observé du système.

Le modèle d'observations est un graphe des marquages accessibles $GA_{OBS}(RdP_{OBS}, M0_{OBS})$ d'un réseau de Petri. RdP_{OBS} est le réseau de Petri du modèle d'observations qui nous est a priori inconnu. En cela, le modèle d'observations ne permet pas de prédire les comportements à observer comme dans les travaux de [Giua, 1997] [Achour, 2005] basés sur un modèle du système. Il nous permet plutôt de modéliser les séquences d'observations sous la forme de toutes les trajectoires pouvant les expliquer.

$M0_{OBS}$ est le marquage initial de ce réseau, qui représente l'état initial du système. Notre modèle d'observations ou graphe d'observations, a donc pour racine l'état initial du système. Au début de la construction, c'est le seul marquage contenu dans l'ensemble des marquages accessibles du graphe.

Dans un cadre de fonctionnement correct et d'un modèle de comportement correct, l'observation devrait se retrouver dans les comportements attendus ($GA_{CA} \supset GA_{OBS}$) puisque MOD_{CA} représente tous les comportements possibles du système.

Etant donné l'impossibilité de déterminer la structure du graphe des marquages du RdP modélisant les observations, à la réception de chaque nouvelle observation nous considérons trois possibilités d'évolution différentes :

– P_1 : le système reste dans son état présent, sans évoluer vers un nouvel état ;

- P_2 : le système évolue vers un état suivant dont l'existence a déjà été mise en évidence par une observation antérieure. L'état initial est considéré comme un état déjà existant et donc potentiellement atteignable par ce type d'évolution ;
- P_3 : le système évolue vers un nouvel état qui n'a jamais été atteint par des observations antérieures, dénoté par M_x.

En considérant ces trois possibilités d'évolutions, nous pouvons représenter l'incertitude dans l'état atteint avec la réception d'une observation, mais également être exhaustifs dans la représentation des évolutions possibles du procédé. Cependant, les possibilités P_1 et P_2 peuvent faire référence à des comportements qui ne sont pas normaux dans le fonctionnement du système.

4.3.2 Construction du MOD_{OBS}

A la réception d'une observation, un ensemble de transitions est mis en place pour représenter les trois possibilités proposées comme le montre la figure 4.3. Toutes ces transitions partent de l'état courant (M_c) dans lequel se trouve MOD_{OBS} et sont dirigées vers différents états comme évoqué précédemment (même état, état antérieur et nouvel état).

La fonction ET_{OBS} étiquette chaque transition créée avec l'événement e_k correspondant à la dernière observation o_k :

$$T_{OBS} \overset{ET_{OBS}}{\longrightarrow} E_{obs}$$

$$t_i \overset{ET_{OBS}}{\rightarrow} ET_{OBS}(t_i) = Eo(o_k)$$

Dans un premier temps, toutes ces évolutions sont représentées dans MOD_{OBS} mais elles vont être affinées dans l'étape de *vérification de cohérence* lors la confrontation de modèles (cette dernière étape sera expliquée postérieurement dans ce manuscrit).

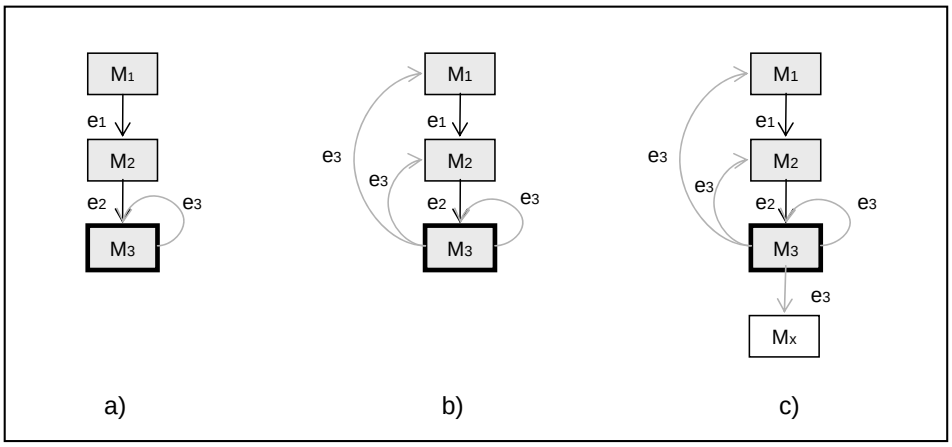


FIG. 4.3 – Ensemble de transitions créé à la réception d'une observation

Dans cette figure, M_3 correspond à l'état courant dans lequel se trouve MOD_{OBS}, après avoir observé et modélisé la séquence $S_{obs} = (o_1, o_2)$ par $S_{ev} = (e_1, e_2)$. M_1 et M_2 représentent les états atteints par e_1 et e_2, respectivement.

Supposons que la dernière observation reçue soit e_3. Dans la figure 4.5(a), nous rajoutons une transition sortant de l'état courant M_3 et arrivant au même état M_3. Cette transition est étiquetée par e_3, et traduit la possibilité d'évolution P_1 i.e. Dans ce cas, l'observation est supposée ne provoquer aucune évolution d'état et le procédé reste dans M_3. La figure 4.5(b) intègre en plus les possibilités d'évolution P_2. Deux transitions étiquetées e_3 et sortant de M_3 sont créées : la première atteint l'état M_2 et la deuxième l'état M_1. Ici le procédé atteint à nouveau, les états qui ont été déjà atteints par d'autres observations de la séquence. Finalement, la figure 4.5(c), intègre la possibilité d'évolution P_3. Pour cela, une transition étiquetée par e_3 et reliant M_3 à un nouvel état noté M_x est rajoutée.

Comme nous pouvons remarquer dans cette démarche, l'espace d'états de ce modèle croit à la réception de chaque nouvelle observation. L'algorithme 4.1 montre la démarche complète de la construction du modèle d'observations. Cet algorithme ne montre pas la détermination du M_x car cette valeur est déterminée postérieurement dans l'étape de *vérification de cohérence*.

L'ordre de traitement P_1, P_2, P_3 choisi dans cet algorithme n'a pas d'importance. Tout autre ordre conduirait aux mêmes résultats.

Algorithme de construction

Soit $M0_{OBS}$ l'état initial du système et M_c son état courant,

1. Initialiser le graphe d'observations $GA_{OBS}(RdP_{OBS}, M0_{OBS})$ avec :
   - l'ensemble des noeuds $M_{OBS} = \{M0_{OBS}\}$
   - l'ensemble d'arcs $T_{OBS} = \phi$
2. A la réception d'une observation o_k
   - Mémoriser l'événement $e_k = Eo(o_k)$ dans la séquence $S_{ev} = (e_1, e_2, \ldots, e_k) = (E_o(o_1), E_o(o_2), \ldots, E_o(o_k))$
   - Construire les transitions :
     - P_1 : créer une transition t_{id} de M_c à M_c étiquetée par e_k
     - P_2 : créer une transition t_{qi} étiquetée par e_k, allant de M_c à chaque sommet contenu dans M_{OBS}, sauf M_c (possibilité P_1
     - P_3 : créer un nouveau noeud M_x et une transition t_n étiquetée par e_k de M_c à M_x
   - Validation des transitions :
     - Appliquer l'algorithme de vérification de cohérence avec MOD_{CA}
     - **Si** la transition t_{id} est vérifiée avec MOD_{CA} **Alors**
       Eliminer les évolutions de type P_2 et P_3 en supprimant les transitions associées
       Mettre à jour l'état courant de GA_{OBS}, $M_c = M_c$
       Aller en 2
     - **Si** une des transitions t_{qi} est vérifiée avec MOD_{CA} **Alors**
       Eliminer les évolutions de type P_1 et P_3 ainsi que celles de type P_2 non validées, en supprimant les transitions associées.
       Mettre à jour l'état courant de GA_{OBS}, $M_c = M_k^1$ où M_k^1 représente l'état courant de MOD_{CA} atteint par l'événement e_k
       Aller en 2
     - **Si** la transition t_n est validée avec MOD_{CA} **Alors**

Eliminer les évolutions de type P_1 et P_2 en supprimant les transitions associées
Déterminer la valeur de M_x, où $M_x = M_g^1$
Mettre à jour l'état courant de GA_{OBS}, $M_c = M_x$
Aller en 2

- **Si** la cohérence est rétablie dans MOD_{CA} **Alors**
  Mettre à jour l'état courant M_c de GA_{OBS} avec $M_c = M_{mod}^1$ où $M_{mod}^1 \in M_{CA}$ sera intégré dans M_x dans le cas d'un marquage nouveau non contenu dans M_{OBS}, $M_c = M_x = M_{mod}^1$
  Eliminer les transitions qui représentent l'événement e_k et qui ne conduisent pas au marquage courant M_c
  Aller en 2
- **Si** la cohérence n'est pas vérifiée **ni avec** MOD_{CA} **ni avec** MOD_C **Alors**
  Impossible de déterminer l'état courant dans GA_{OBS}

3. fin de l'algorithme

Ainsi le modèle d'observations est un graphe formé par un ensemble de sommets et un ensemble d'arcs orientés. Les sommets de ce graphe représentent les états du système atteints à partir du marquage initial par une séquence d'observations. Les arcs représentent les événements observés qui font passer le système d'un état vers un autre. Un arc va donc relier deux sommets M_c et M_x s'il existe un événement observé pour les lier. Dans cet algorithme, M_{mod}^1 mémorise l'état courant lorsque la cohérence est rétablie.

Nous allons maintenant présenter la détection, réalisée par la vérification de cohérence entre MOD_{OBS} et les modèles du système, MOD_{CA} et MOD_C.

4.4 La vérification de la cohérence

Comme nous l'avons exposé plus tôt l'étape de détection nécessite de vérifier :

- la cohérence entre MOD_{OBS} et le modèle de comportement attendu MOD_{CA}
- la cohérence entre MOD_{OBS} et le modèle de la commande MOD_C

Ces vérifications sont réalisées par synchronisation de comportements. A la réception d'un événement, s'il est possible de synchroniser le comportement observé avec un comportement du modèle du comportement attendu, cela signifie que le modèle de comportement attendu est cohérent avec le fonctionnement observé. Dans le cas contraire, le modèle du comportement attendu est jugé incohérent. Le raisonnement est le même avec le modèle de la commande.

La figure 4.4 représente les interactions entre l'algorithme de construction du modèle d'observations, les algorithmes de vérification de cohérence, l'algorithme d'identification de la configuration (qui sera présenté dans le chapitre 5), ainsi qu'avec l'algorithme de modification de modèles qui sera décrit dans le chapitre 6.

4.4.1 Vérification de cohérence entre MOD_{OBS} et MOD_{CA}

Pour une séquence d'observations donnée, l'ensemble des sommets (i.e. marquages) de MOD_{CA} peut être divisé en deux : l'ensemble des marquages atteints ou visités avec la séquence d'observations note M_F^1 et l'ensemble des marquages qui ne sont pas atteints avec cette séquence d'observations, noté M_{NF}^1. La figure 4.5 nous montre cette partition des états de MOD_{CA}.

Soit M_k^1 le marquage courant dans MOD_{CA}. A la réception de l'événement e_k pour que la synchronisation soit possible il doit exister dans MOD_{CA} une transition franchissable, étiquetée avec l'événement observé e_k, permettant d'atteindre :

- soit le marquage courant M_k^1
- soit un autre marquage M_h^1 déjà visité, $M_h^1 \in M_F^1$
- soit un autre marquage M_g^1 non encore visité, $M_g^1 \in M_{NF}^1$

La recherche d'une telle transition va être réalisée en mettant en place le produit cartésien de MOD_{CA} et MOD_{OBS}. Ce produit cartésien nous donne le graphe de vérification noté $GVER^{CA}$ défini par :

- un ensemble de sommets : $M_{VER} = \{M/\sigma \in T_{OBS}^* \cup T_{CA}^* \, t.q. \, M0_{OBS\times CA} \xrightarrow{\sigma} M\}$ Un sommet M est l'association d'un sommet de MOD_{CA} et d'un sommet de MOD_{OBS}, on notera $M = (M1, M2)$ avec $M1 \in M_{OBS}$ et $M2 \in M_{CA}$.
- un ensemble d'arcs orientés noté A_{VER}. Un arc relie un sommet M à un sommet M' seulement s'il existe une transition t franchissable permettant de passer du marquage M au marquage M' ;

Trois types de transitions peuvent être présents dans le graphe de vérification :

1. *(M1, M2)* $\xrightarrow{t}$ *(M'1, M2)*
2. *(M1, M2)* $\xrightarrow{t}$ *(M1, M'2)*
3. *(M1, M2)* $\xrightarrow{t}$ *(M'1, M'2)*

Une transition de type 1, indique qu'il y a seulement une évolution dans MOD_{OBS}. Ceci signifie qu'à partir du marquage courant $(M1, M2)$, la transition t n'est franchissable que dans MOD_{OBS}.

Une transition de type 2 traduit le cas contraire : la transition t n'est franchissable que dans MOD_{CA}.

Enfin, une transition de type 3 définit des comportements synchrones i.e la transition t est franchissable dans les deux modèles. Les valeurs des marquages de départ pour chacun des modèles sont identiques. A partir de ces marquages la transition t est franchissable et conduit vers des marquages identiques dans les deux modèles. De cette manière, il nous est possible d'identifier les sommets inconnus notés M_x du modèle d'observations. En effet, nous allons considérer que cet état inconnu est identique à l'état atteint dans MOD_{CA} par le tir de t et que cet état est un état non visité ($M_g^1 \in M_{NF}^1$).

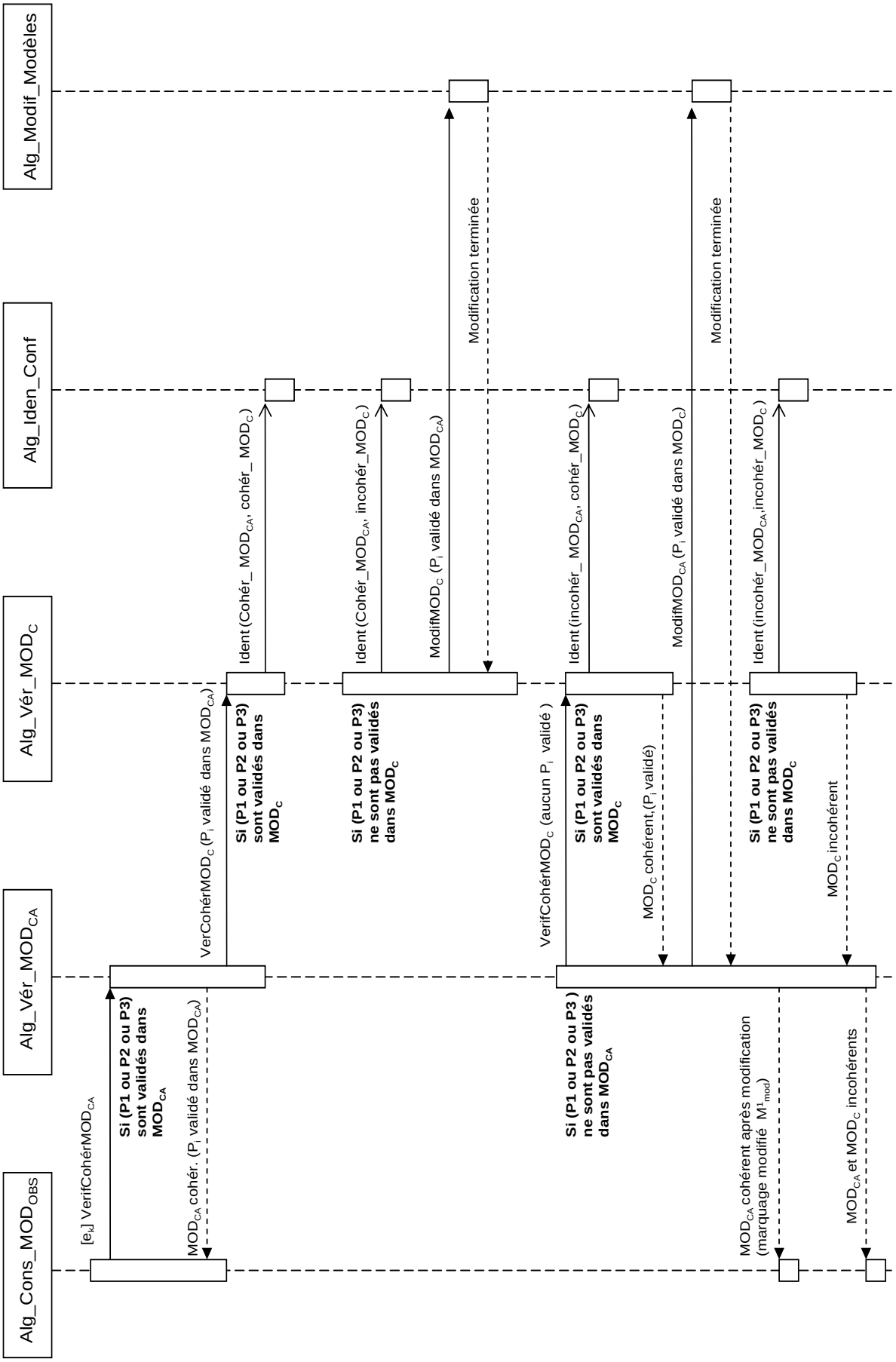


FIG. 4.4 – Interactions entre les algorithmes pour la vérification de la cohérence

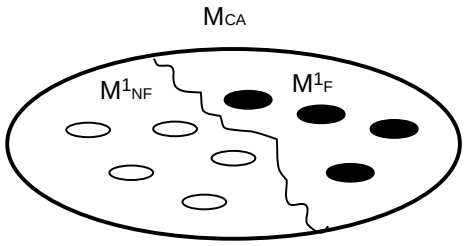


FIG. 4.5 – Partition de M_{CA} l'ensemble des états de MOD_{CA}

La synchronisation des modèles a donc lieu si l'existence d'une transition de type 3 est vérifiée au niveau du graphe de vérification. Ce type de transition sera représenté par : $(t_i(e_k)^{OBS}, t_j(e_k)^{CA})$ avec t_i la transition franchissable dans MOD_{OBS}, associée à l'événement e_k et t_j la transition franchissable dans MOD_{CA} associée à l'événement e_k

Or, une transition de type 3 peut correspondre à différentes évolutions au niveau des marquages des modèles comme nous l'avons vu dans le cas du modèle d'observations. En effet, il se peut qu'au niveau des modèles le marquage atteint soit un marquage déjà visité, ou un nouveau marquage ou bien encore que le marquage atteint soit le marquage courant.

Nous pouvons donc conclure, qu'à partir des marquages courants de MOD_{OBS} et MOD_{CA} (notés M_c et M_k^1) et pour un événement observé e_k, l'existence de comportements synchronisés dépend de l'existence d'une transition t, étiquetée par l'événement observé (e_k) et franchissable telle que :

– le tir de t ne modifie pas le marquage courant de MOD_{CA}.

$$\exists t \in T_{CA} \ \mid \ M_k^1 \xrightarrow{t} M_k^1$$

– le tir de t conduit vers un état déjà visité.

$$\exists t \in T_{CA} \ \mid \ M_k^1 \xrightarrow{t} M_h^1 \ avec \ M_h^1 \in M_F^1$$

– le tir de t conduit vers un état qui n'a pas encore été visité.

$$\exists t \in T_{CA} \ \mid \ M_k^1 \xrightarrow{t} M_g^1 \ avec \ M_g^1 \in M_{NF}^1$$

S'il existe dans MOD_{CA} une transition d'un de ces trois types, dans l'étape de vérification nous allons retrouver des comportements synchronisés et les deux modèles MOD_{OBS} et MOD_{CA} seront déclarés cohérents.

Il est toutefois nécessaire de remarquer que pour certains systèmes dans les deux premiers cas, bien que la cohérence soit vérifiée le comportement du système peut être anormal dans la mesure où les états atteints par le tir de la transition t peuvent traduire des comportements incorrects du système. Prenons pour exemple un système dont le

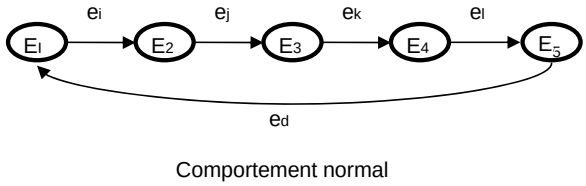


FIG. 4.6 – Comportement cyclique d'un système

comportement est décrit figure 4.6. Il s'agit d'un comportement purement cyclique décrit par l'état initial E_I et par un ensemble d'états intermédiaires.

Les évolutions représentées sur la figure 4.7 ne sont donc pas autorisées et sont assimilées à une défaillance au niveau du système. Si ces évolutions sont représentées dans MOD_{CA}, il ne sera pas possible de détecter les défaillances associées, car il n'y aura pas d'incohérence entre les modèles, lorsque ces comportement seront observés.

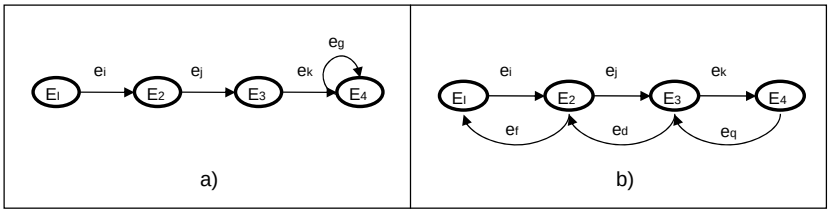


FIG. 4.7 – Représentation d'une défaillance potentielle dans le système

Compte tenu des résultats précédents l'algorithme de vérification de la cohérence entre MOD_{OBS} et MOD_{CA} est le suivant :

Algorithme de vérification de cohérence entre MOD_{OBS} et MOD_{CA}

1. Soit e_k le dernier événement observé
2. Soit MOD_{OBS} le modèle d'observation obtenu après la réception de e_k
3. Soit MOD_{CA} le modèle du comportement attendu
4. Soit $GVER^{CA}$ le graphe de vérification obtenu à partir de MOD_{OBS} et MOD_{CA},
5. **Si** au niveau de $GVER^{CA}$ à partir de l'état (M_c, M_k^1), il existe une transition t du type $(t_i(e_k)^{OBS}, t_j(e_k)^{CA})$ traduisant des comportements synchronisés **Alors**
   (a) Vérifier la cohérence par rapport à des comportements de type P_1 de MOD_{OBS} :
   Si t est telle que $(M_c, M_k^1) \xrightarrow{t} (M_c, M_k^1)$ **Alors**
   - Le comportement de type P_1 est vérifié
   - Le comportement capturé par MOD_{CA} est cohérent avec le comportement observé
   - Faire évoluer MOD_{CA} par le franchissement de la transition sensibilisée t associée à l'événement observé e_k.
   - Mettre à jour l'état courant de MOD_{CA}, avec $M_k^1 = M_k^1$,
   - Transmettre le résultat à l'algorithme de construction de MOD_{OBS}
   - Appliquer le résultat à l'algorithme de vérification de cohérence entre MOD_{OBS} et MOD_C à partir de ce résultat

- Aller en 16

(b) Vérifier la cohérence par rapport à des comportements de type P_2 de MOD_{OBS} :

Si t est telle que $(M_c, M_k^1) \xrightarrow{t} (M_*, M_h^1)$ avec $M_h^1 \in M_F^1$, $M_* \in M_{OBS}$, M_* déjà visité et $M_* \neq M_c$ **Alors**

- Le comportement de type P_2 est vérifié
- Le comportement capturé par MOD_{CA} est cohérent avec le comportement observé
- Faire évoluer MOD_{CA} par le franchissement de la transition sensibilisée t associée à l'événement observé e_k.
- Mettre à jour l'état courant de MOD_{CA}, avec $M_k^1 = M_h^1$,
- Transmettre le résultat à l'algorithme de construction de MOD_{OBS}
- Appliquer le résultat à l'algorithme de vérification de cohérence entre MOD_{OBS} et MOD_C à partir de ce résultat
- Aller en 16

(c) Vérifier la cohérence par rapport à des comportements de type P_3 de MOD_{OBS} :

Si t est telle que $(M_c, M_k^1) \xrightarrow{t} (M_x, M_g^1)$ avec $M_g^1 \in M_{NF}^1$ **Alors**

- Le comportement de type P_3 est vérifié
- Le comportement capturé par MOD_{CA} est cohérent avec le comportement observé
- Faire évoluer MOD_{CA} par le franchissement de la transition sensibilisée t associée à l'événement observé e_k.
- Mettre à jour l'état courant de MOD_{CA}, avec $M_k^1 = M_g^1$,
- Transmettre la valeur M_g^1 et le résultat à l'algorithme de construction de MOD_{OBS}
- Appliquer le résultat à l'algorithme de vérification de cohérence entre MOD_{OBS} et MOD_C à partir de ce résultat
- Aller en 16

6. Le comportement capturé par MOD_{CA} est incohérent avec le comportement observé
7. Appliquer l'algorithme de vérification de cohérence entre MOD_{OBS} et MOD_C
8. **Si** la cohérence est vérifiée avec MOD_C **Alors**
   - Appliquer l'algorithme de modification de MOD_{CA} basé sur le comportement vérifié dans MOD_C
   - Faire évoluer MOD_{CA} par le franchissement de la transition sensibilisée t associée à l'événement observé e_k.
   - Mettre à jour l'état courant de MOD_{CA}, avec $M_k^1 = M_{mod}^1$ où M_{mod}^1 dénote le marquage atteint par le franchissement de t.
   - Transmettre le résultat à l'algorithme de construction de MOD_{OBS}
   - Aller en 1.
9. L'ensemble d'états possibles $Eep = \{M_j \mid M_j \in M_{GA_{CA} \backslash\backslash GA_C}\}$
10. Attendre le prochain événement e_k
11. **Si** $\nexists t \in Eep$ telle que $Ev(t) = e_k$ **Alors**

- L'état courant dans GA_{CA} est inconnu
- L'état courant dans GA_C est inconnu
- Le rétablissement de la cohérence est impossible

12. **Si** $\exists! t \in Eep$ telle que $Ev(t) = e_k$ **Alors**
    - L'état courant dans GA_{CA} est déterminé
    - L'état courant dans GA_C est déterminé
    - Le rétablissement de la cohérence est possible par la modification de MOD_{CA} et MOD_C, basée sur le premier événement de la séquence d'événements observés
    - Aller en 1
13. L'ensemble d'états résultants $Eer = \{M_j \mid \exists M_i \in Eep \wedge M_i \stackrel{t}{\longmapsto} M_j \text{ avec } Ev(t) = e_k\}$
14. $Eep \leftarrow Eer$
15. Aller en 10
16. Fin de l'algorithme.

En cas d'incohérence entre les deux modèles le système de surveillance va déclencher l'algorithme d'identification de configuration. Une fois l'identification terminée l'algorithme de modification du modèle est lancé pour rétablir la cohérence. Ces deux algorithmes font l'objet des chapitres suivants (5, 6).

Lorsque la cohérence est rétablie MOD_{CA} évolue pour mettre à jour son état conformément à l'observation. L'état courant est déterminé et mémorisé en M^1_{mod}. Cet état est communiqué au modèle d'observation MOD_{OBS} et va être le point de départ pour la modélisation de la prochaine observation.

Si une observation peut être trouvée cohérente avec le modèle de comportement attendu, rien ne garantit qu'elle soit aussi cohérente avec le modèle de la commande. Il est donc nécessaire de vérifier également la cohérence entre le modèle d'observations et le modèle de la commande : quand MOD_{CA} est cohérent avec le comportement observé nous utilisons le type de comportement (P_1, P_2 ou P_3) vérifié dans ce modèle pour orienter la vérification de la cohérence dans MOD_C. Si par exemple, la cohérence est vérifiée dans MOD_{CA} par une évolution du type P_1, alors ce même type d'évolution doit pouvoir être validé dans MOD_C.

Quand il y a incohérence entre MOD_{OBS} et MOD_{CA} nous appliquons l'algorithme de vérification de cohérence pour MOD_C et nous attendons le résultat de cette vérification. Dans le cas où le résultat indique qu'il existe cohérence entre MOD_{OBS} et MOD_C nous utilisons le type de comportement vérifié pour guider les modifications du MOD_{CA}. Dans le cas contraire où aucun type de comportement du MOD_{OBS} est cohérent avec MOD_C, nous devons attendre de nouvelles observations pour pouvoir effectuer l'étape de modification de MOD_{CA} et MOD_C car nous n'avons pas de moyens pour guider la modification.

Nous décrivons maintenant le principe de vérification de cohérence du MOD_C.

4.4.2 Vérification de cohérence entre MOD_{OBS} et MOD_C

Il s'agit de vérifier que l'état attendu par le modèle de la commande est bien celui observé. Cette vérification va donc naturellement s'appuyer sur le modèle d'observations qui capture les comportements observés.

Les états des deux modèles mis en jeu dans cette vérification, n'ont pas de correspondance directe, contrairement à ce qui était le cas avec MOD_{CA}. Le modèle de la commande modélise la séquence d'activités à réaliser et les moyens de production à utiliser. Chaque activité décrite dans le modèle correspond à une séquence de commandes envoyée au procédé. C'est l'exécution de cette séquence de commande qui va provoquer le comportement observé. Le modèle d'observations lui, modélise directement le comportement observé.

Néanmoins, ces deux modèles sont liés par les comportements observables. Ces comportements qui sont l'objet de la modélisation dans MOD_{OBS} sont associés au niveau de la commande aux compte-rendus d'exécution.

Quand une séquence de commande est envoyée au procédé, le modèle de la commande reste dans un état qui indique que cette séquence est en cours d'exécution comme le montre la figure 4.8. Quand la commande reçoit le compte-rendu d'exécution, la séquence est considérée comme terminée et une nouvelle séquence de commandes est envoyée.

L'observation d'un événement va donc nous indiquer la fin de l'exécution d'une commande et en même temps l'état dans lequel se trouve le procédé après l'exécution de cette commande.

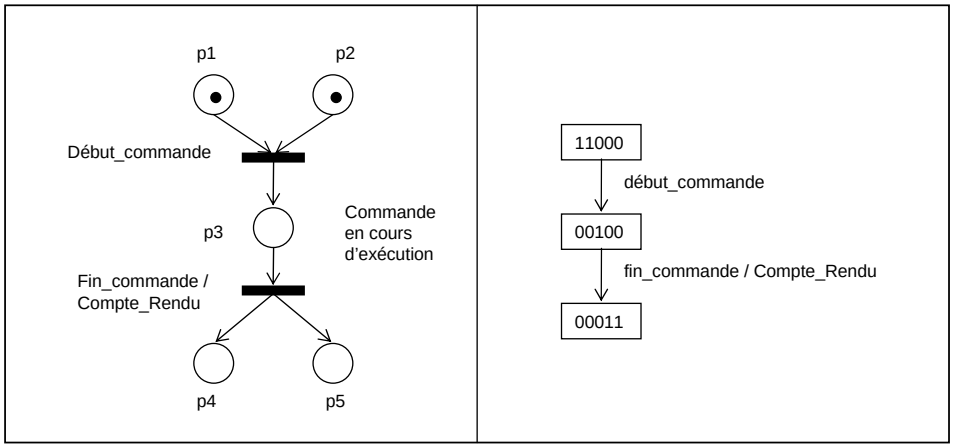


FIG. 4.8 – Modèle de la commande

De la même manière que dans le cas de la vérification de cohérence de MOD_{CA}, la vérification de cohérence pour MOD_C est effectuée en réalisant le produit cartésien entre MOD_{OBS} et MOD_C. Le résultat est mis en place dans un graphe de vérification $GVER^C$. La recherche de comportement synchronisé dans $GVER^C$ est utilisée pour vérifier la cohérence de MOD_C. Ce graphe est défini sur le même principe que le graphe de vérification $GVER^{CA}$ (section 4.4.1 de la page 75).

Nous utilisons le résultat de la vérification de MOD_{CA} pour guider notre recherche de comportements synchronisés, c'est-à-dire que nous cherchons s'il existe dans MOD_C

un comportement du même type que celui vérifié dans MOD_{CA}, car c'est le seul comportement du modèle d'observations permettant d'intégrer l'événement e_k. Si MOD_{CA} est incohérent avec MOD_{OBS} alors nous effectuons la vérification sur les trois types de comportements décrits dans MOD_{OBS} pour chercher des comportements synchronisés de n'importe quel type.

Comme dans le cas du modèle de comportement attendu MOD_{CA}, l'information émanant de cette incohérence est utilisée pour identifier la configuration entre le modèle d'observations et le modèle de la commande. La déclaration d'une incohérence dans MOD_C, déclenche la fonction de diagnostic (décrite dans le chapitre 6) dont l'objectif est de rétablir la cohérence entre le modèle de la commande et le modèle d'observations. Ce rétablissement de la cohérence va être réalisé en modifiant MOD_C. Une fois la modification réalisée la transition associée à l'événement observé e_k désormais sensibilisée, est franchie pour mettre à jour l'état du modèle. Nous allons utiliser le même principe que celui utilisé dans MOD_{CA} pour diviser les états de la commande. Donc ici M^2_{NF} et M^2_F indiquent les états encore non visités et les états visités dans le modèle de la commande, respectivement. L'algorithme suivant donne la procédure pour la vérification de la cohérence entre MOD_{OBS} et MOD_C.

Algorithme de vérification de cohérence entre MOD_{OBS} et MOD_C

1. Soit e_k le dernier événement observé
2. Soit Sc la dernière séquence de commande envoyée vers le procédé
3. Soit MOD_C le modèle de la commande et M^2_k l'état courant du modèle de la commande
4. Attendre le résultat de vérification de cohérence de MOD_{OBS} et MOD_{CA}
5. Soit $GVER^C$ le graphe de vérification obtenu à partir de MOD_{OBS} et MOD_C,
6. **Si** au niveau de $GVER^C$ à partir de l'état (M_c, M^2_k) il existe une transition t du type $(t_i(ek)^{OBS}, t_j(ek)^C)$ Alors

   (a) **Si** la cohérence avec MOD_{CA} a été vérifiée par validation d'une évolution de type P_1 **Alors** vérifier la cohérence par rapport à P_1 de MOD_{OBS} :

   Si t est telle que $(M_c, M^2_k) \xrightarrow{t} (M_c, M^2_k)$ **Alors**
   - Le comportement de type P_1 est vérifié, le comportement capturé par MOD_C est cohérent avec le comportement observé
   - Transmettre à l'algorithme d'identification de configuration, le résultat de la cohérence avec MOD_{CA} et avec MOD_C
   - Faire évoluer MOD_C par le franchissement de la transition sensibilisée t associée à l'événement observé e_k et mettre à jour l'état courant de MOD_C, avec $M^2_k = M^2_k$.
   - Aller en 12

   Sinon
   - Aller en 7

   (b) **Si** la cohérence avec MOD_{CA} a été vérifiée par validation d'une évolution de type P_2 **Alors** vérifier la cohérence par rapport à P_2 de MOD_{OBS} :

   Si t est telle que $(M_c, M^2_k) \xrightarrow{t} (M_*, M^2_h)$ avec $M^2_h \in M^2_F$, $M_* \in M_{OBS}$, M_* déjà visité et $M_* \neq M_c$ **Alors**

- Le comportement de type P_2 est vérifié, le comportement capturé par MOD_C est cohérent avec le comportement observé
- Transmettre à l'algorithme d'identification de configuration, le résultat de la cohérence avec MOD_{CA} et avec MOD_C
- Faire évoluer MOD_C par le franchissement de la transition sensibilisée t associée à l'événement observé e_k et mettre à jour l'état courant de MOD_C, avec $M_k^2 = M_h^2$,
- Aller en 12

Sinon

- Aller en 7

(c) **Si** la cohérence avec MOD_{CA} a été vérifiée par validation d'une évolution de type P_3 **Alors** vérifier la cohérence par rapport à P_3 de MOD_{OBS} :

Si t est telle que $(M_c, M_k^2) \xrightarrow{t} (M_x, M_g^2)$ avec $M_g^2 \in M_{NF}^2$ **Alors**

- Le comportement de type P_3 est vérifié, le comportement capturé par MOD_C est cohérent avec le comportement observé
- Transmettre à l'algorithme d'identification de configuration, le résultat de la cohérence avec MOD_{CA} et avec MOD_C
- Faire évoluer MOD_C par le franchissement de la transition sensibilisée t associée à l'événement observé e_k et mettre à jour l'état courant de MOD_C, avec $M_k^2 = M_g^2$,
- Aller en 12

Sinon

- Aller en 7

(d) **Si** le comportement a été incohérent avec MOD_{CA} **Alors** vérifier la cohérence par rapport à des évolutions de type P_1, P_2 et P_3 de MOD_{OBS} :

– **Si** t est telle que $(M_c, M_k^2) \xrightarrow{t} (M_c, M_k^2)$ **Alors**

- Le comportement de type P_1 est vérifié, le comportement capturé par MOD_C est cohérent avec le comportement observé
- Transmettre à l'algorithme d'identification de configuration, le résultat de l'incohérence avec MOD_{CA} et la cohérence avec MOD_C
- Faire évoluer MOD_C par le franchissement de la transition sensibilisée t associée à l'événement observé e_k et mettre à jour l'état courant de MOD_C, avec $M_k^2 = M_k^2$,
- Transmettre le résultat à l'algorithme de vérification de cohérence de MOD_{CA}
- Aller en 12

– **Si** t est telle que $(M_c, M_k^2) \xrightarrow{t} (M_*, M_h^2)$ avec $M_h^2 \in M_F^2$, $M_* \in M_{OBS}$, M_* déjà visité et $M_* \neq M_c$**Alors**

- Le comportement de type P_2 est vérifié, le comportement capturé par MOD_C est cohérent avec le comportement observé
- Transmettre à l'algorithme d'identification de configuration, le résultat de l'incohérence avec MOD_{CA} et la cohérence avec MOD_C
- Faire évoluer MOD_C par le franchissement de la transition sensibilisée t associée à l'événement observé e_k et mettre à jour l'état courant de MOD_C, avec $M_k^2 = M_h^2$,
- Transmettre le résultat à l'algorithme de vérification de cohérence de

MOD_{CA}
- Aller en 12

– **Si** t est telle que $(M_c, M_k^2) \xrightarrow{t} (M_x, M_g^2)$ avec $M_g^2 \in M_{NF}^2$ **Alors**
- Le comportement de type P_3 est vérifié, le comportement capturé par MOD_C est cohérent avec le comportement observé
- Transmettre à l'algorithme d'identification de configuration, le résultat de l'incohérence avec MOD_{CA} et la cohérence avec MOD_C
- Faire évoluer MOD_C par le franchissement de la transition sensibilisée t associée à l'événement observé e_k et mettre à jour l'état courant de MOD_C, avec $M_k^2 = M_g^2$,
- Transmettre le résultat à l'algoritme de vérification de cohérence de MOD_{CA}
- Aller en 12

Sinon
- Transmettre à l'algorithme d'identification de configuration, le résultat de l'incohérence avec MOD_{CA} et l'incohérence avec MOD_C
- Transmettre le résultat à l'algorithme de vérification de cohérence de MOD_{CA}
- Aller en 12

7. Le comportement capturé par MOD_C est incohérent avec le comportement observé

8. Transmettre à l'algorithme d'identification de configuration, le résultat de la cohérence avec MOD_{CA} et l'incohérence avec MOD_C
9. Appliquer l'algorithme de modification pour modifier MOD_C en s'appuyant sur le comportement vérifié dans MOD_{CA}
10. Faire évoluer MOD_C par le franchissement de la transition sensibilisée t associée à l'événement observé e_k et déterminer .
11. Mettre à jour l'état courant de MOD_C, avec $M_k^2 = M_{mod}^2$ où M_{mod}^2 dénote le marquage atteint pour le franchissement de t.
12. Fin de l'algorithme

Si pour un événement observé e_k la cohérence n'est pas vérifiée dans MOD_{CA} et MOD_C l'étape de modification ne pourra pas être appliquée directement suite à la détection et ceci par manque d'informations pertinentes pour guider la modification de modèles.

C'est un cas d'impasse similaire à celui d'un échec de diagnostic suivant la détection d'un symptôme de défaillance.

Conclusions

Nous avons décrit dans ce chapitre la phase de détection d'incohérence de notre approche. Il s'agit de la rupture de la cohérence entre la réalité et la référence. Nous utilisons deux modèles comme référence de bon comportement : le premier modèle traduit uniquement le comportement attendu (observable) et est appelé MOD_{CA}, le deuxième modèle représente la façon dont le procédé doit être utilisé pour répondre à un ensemble de requêtes de l'utilisateur du système et est appelé MOD_C.

Ce chapitre explique également comment les observations issues du fonctionnement du procédé sont capturées dans un modèle d'observations appelé MOD_{OBS}. Les mécanismes de construction de MOD_{OBS} ont été détaillés. C'est ce modèle d'observations qui est utilisé pour faire les comparaisons entre le comportement observé et le comportement décrit dans les modèles du système.

Ces trois modèles sont implémentés en utilisant le graphe des marquage accessibles des réseaux de Petri.

Nous avons également montré que cette étape de détection est réalisée par la vérification de la cohérence entre les modèles. Cette vérification est réalisée en effectuant le produit cartésien des modèles. Si ce produit synchrone de modèles ne couvre pas le comportement observé, alors nous déclarons une incohérence entre les modèles, traduisant un symptôme de défaillance ou une erreur de modèle.

Traditionnellement, l'étape suivant la détection est l'étape du diagnostic de la défaillance. Mais étant donnée notre hypothèse de départ sur la possibilité d'erreurs dans les modèles du système, l'étape suivante de notre approche est l'identification de ces erreurs en terme de configurations générées entre les ensembles de trajectoires des modèles MOD_{CA} et MOD_C.

Chapitre 5

Identification de la configuration des ensembles de trajectoires des modèles du système

Introduction

Étant donné que dans notre approche de diagnostic nous envisageons la possibilité d'erreurs dans les modèles, à la détection d'une incohérence, nous ne pouvons pas être sûrs qu'il s'agit d'un symptôme de défaillance ou d'un faux symptôme résultant d'une erreur dans un modèle. Pour lever cette incertitude nous allons introduire dans ce chapitre un troisième modèle MOD_{CR} du système qui va nous servir pour distinguer ces deux situations. Ce modèle s'ajoute aux deux déjà présentés dans le chapitre 4 : (MOD_{CA} et MOD_C). Bien sûr, indépendamment des modèles du système, le modèle d'observation MOD_{OBS} reste avec le même rôle que précédemment.

Pour distinguer les faux symptômes des symptômes de défaillances, il est nécessaire de connaître exactement comment les différents modèles se situent les uns par rapport aux autres. L'ajout d'un troisième modèle va compliquer notablement cette identification, mais c'est le passage obligé pour espérer lever l'incertitude faux symptôme /symptôme de défaillance. L'étude que nous présentons propose d'identifier progressivement les recouvrements existants entre les modèles, ceci dans le but de savoir quel(s) modèle(s) doi(ven)t être modifié(s) lors de la détection d'un faux symptôme.

La technique présentée dans ce chapitre a un double but : la première c'est d'identifier la manière dont les modèles sont situés l'un par rapport à l'autre, la deuxième c'est lever une partie de l'incertitude : défaillance ou erreur dans le modèle.

Tout d'abord nous présentons les Configurations des Ensembles de Trajectoires (CET) des modèles du système, par la suite, par souci de simplicité, nous utiliserons le terme configuration. Dans un premier temps, nous illustrons nos propos avec deux modèles uniquement MOD_{CA}, MOD_C car il est possible dans ce cas d'énumérer exhaustivement les différentes configurations. Nous appliquerons le même raisonnement au cas des trois modèles où le troisième modèle est MOD_{CR}.

5.1 Configuration de deux modèles : A et C

Pour introduire les concepts nécessaires à notre raisonnement et rester dans un cadre général, nous allons supposer l'existence de deux modèles A et C décrivant chacun une vue du comportement normal du procédé. L'incohérence va être mise en lumière lorsqu'une observation o_i est associée à un événement $Eo(o_i)$ qui ne correspond à aucun événement attendu par au moins l'un des modèles A et C.

Définition d'une configuration entre deux modèles

Nous avons vu dans le dernier chapitre qu'il est possible qu'un comportement observé soit incohérent avec un seul des modèles (A ou C), même si tous les deux sont supposés modéliser les comportements normaux du système. Nous avons évoqué aussi que la cause de cette situation sont les erreurs possibles dans les modèles, erreurs engendrées pendant la phase de modélisation du système.

Une incohérence entre deux modèles vis-à-vis d'une trajectoire

$$T_R = (M_1, e_1, M_2, e_2, \ldots, e_{n-1}, M_n)$$

apparaît lorsque la séquence d'événements sous-jacente

$$S_{ev} = (e_1, e_2, \ldots, e_{n-1})$$

est acceptée par un seul modèle.

La trajectoire T_R peut être décomposée en un préfixe T_{R1} et un suffixe T_{R2} tels que

$$T_{R1} = (M_1, e_1, M_2, e_2, \ldots, e_{j-1}, M_j)$$

soit une trajectoire cohérente avec les deux modèles et que

$$T_{R2} = (M_j, e_j, M_{j+1}, e_{j+1}, \ldots, e_{n-1}, M_n)$$

ne soit consistante qu'avec un seul. Ceci met en évidence que T_{R_1} est une trajectoire appartenant aux deux modèles alors que T_{R2} n'appartient qu'à l'un d'eux.

La décomposition d'une trajectoire T_R en T_{R1} et T_{R2} possédant les propriétés ci-dessus caractérise l'intersection entre les ensembles de trajectoires des modèles A et C. Si $TR(X)$ est l'ensemble des trajectoires contenues dans un modèle X et que T_{R2} ne soit pas reconnue par C (sans perte de généralité), alors on a

$$T_{R1} \in TR(A) \cap TR(C) \text{ et } T_{R2} \in TR(A) \backslash (TR(A) \cap TR(C))$$

Cette définition est à mettre en regard avec le principe de la détection décrit au chapitre précédent. Dans notre présentation, seul le dernier événement e_{n-1} est responsable de l'incohérence et la configuration sera caractérisée par $T_{R2} = (M_{n-1}, e_{n-1}, M_n)$ la dernière transition du modèle d'observation MOD_{OBS} non acceptée par les deux modèles. Chaque fois que l'un des modèles permettra de détecter une incohérence, la connaissance sur les intersections entre ces ensembles progressera éventuellement. C'est cette connaissance qui nous permettra de caractériser par la suite les symptômes faux ou de défaillance.

5.1.1 Configurations possibles avec deux modèles

La taille du problème est ici réduite et nous permet de l'aborder par énumération des solutions possibles. Il est en effet assez simple de trouver les différentes configurations des ensembles $TR(A)$ et $TR(C)$. Si nous convenons de noter :

A l'ensemble $TR(A)\backslash(TR(A) \cap TR(C))$,

C l'ensemble $TR(C)\backslash(TR(A) \cap TR(C))$,

AC l'ensemble $TR(A) \cap TR(C)$,

il suffit d'énumérer les huit configurations en considérant l'existence de chaque ensemble (A, C, AC). Nous avons donc 8 configurations notées :

ϕ : configuration $(\overline{A}, \overline{C}, \overline{AC})$ dans laquelle n'apparaît aucun des trois ensembles.

$\{A\}$: configuration $(A, \overline{C}, \overline{AC})$ dans laquelle n'apparaît que l'ensemble A,

$\{C\}$: configuration $(\overline{A}, C, \overline{AC})$ dans laquelle n'apparaît que l'ensemble C,

$\{AC\}$: configuration $(\overline{A}, \overline{C}, AC)$ dans laquelle A est égal à C,

$\{A, AC\}$: configuration $(A, \overline{C}, AC)$ dans laquelle C est inclus dans A,

$\{C, AC\}$: configuration $(\overline{A}, C, AC)$ dans laquelle A est inclus dans C,

$\{A, C\}$: configuration $(A, C, \overline{AC})$ dans laquelle A et C sont disjoints,

$\{A, C, AC\}$: configuration (A, C, AC) dans laquelle A et C possèdent une intersection sans qu'il n'y ait inclusion de l'un dans l'autre.

Ces huit configurations sont construites en considérant chaque élément A, C et AC comme une variable booléenne qui apparaît ou non dans la configuration. Avec k éléments, nous aurions donc 2^k configurations.

Comme nous l'avons indiqué, si les deux modèles existent réellement (dans notre approche ce sont les modèles de bon fonctionnement) alors les configurations ϕ, A et C n'ont aucun sens. Nous éliminons donc les configurations qui ne contiennent pas explicitement les deux symboles A et C.

5.1.2 Analyse des configurations valides dans notre approche

Pour passer à l'étude avec trois modèles, nous devons formaliser le mécanisme d'éliminations car une approche énumérative est trop complexe.

Nous venons d'éliminer :

1. l'ensemble vide dans lequel n'apparaît aucun des deux modèles,
2. les ensembles formés par un seul élément, sauf l'ensemble formé par l'élément AC qui à lui seul couvre bien les deux ensembles de départ.

Les sous-ensembles qui peuvent être utilisés pour former des configurations, doivent contenir des éléments représentant les deux modèles. En appliquant cette procédure nous obtenons 5 configurations :

$$\{AC\}, \{A, C\}, \{A, AC\}, \{C, AC\}, \{A, C, AC\}$$

La figure 5.1 illustre les cinq configurations des ensembles de trajectoires l'un par rapport à l'autre.

On peut en déduire les configurations correspondantes.

- Dans la figure 5.1.a les ensembles sont disjoints. Toute trajectoire sera incohérente avec au moins un des deux modèles. Dans ce cas $T_{R2} = T_R$ et T_{R1} est vide.
- Dans la figure 5.1.b il existe une intersection entre les deux ensembles mais aucun n'est inclus dans l'autre. C'est le cas le plus courant où les mêmes comportements normaux apparaissent dans les deux modèles qui contiennent toutefois chacun des comportements spécifiques.
- Dans la figure 5.1.c les deux ensembles sont identiques. C'est le cas idéal puisque tous les comportements normaux sont décrits dans les deux modèles.
- Dans les figures 5.1.d et 5.1.e, un des ensembles est inclus dans l'autre. L'incohérence de l'observation avec les trajectoires de l'ensemble incluant, implique dans ce cas là, l'incohérence avec l'autre ensemble de trajectoire.

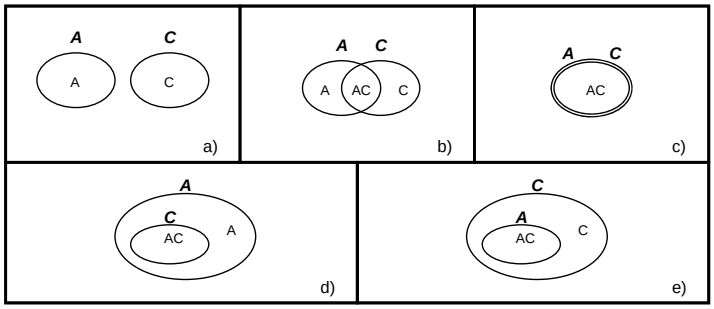


FIG. 5.1 – Configurations pour deux modèles

Maintenant, supposons que le modèle A représente le modèle de comportement attendu MOD_{CA}, et que le modèle C représente le modèle de la commande, MOD_C. Ces 5 configurations sont interprétées et présentées ci-après.

1. Dans la première configuration aucun des comportements décrits dans le MOD_{CA} n'est retrouvé dans le modèle de la commande MOD_C. Il s'agit d'une configuration quelque peu absurde dans le sens où l'analyse a priori montrerait immédiatement l'incohérence complète de ces deux modèles. Graphiquement cette configuration est montrée par la figure 5.2.

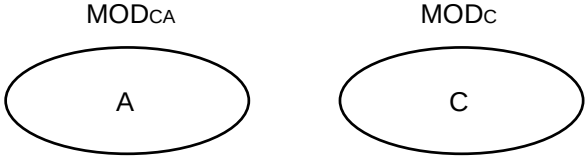


FIG. 5.2 – Modèles disjoints

2. Dans la deuxième configuration, il existe une intersection des ensembles de comportements des deux modèles, autrement dit, il existe des comportements dans MOD_{CA} appartenant également à MOD_C. Cependant, il existe des comportements dans MOD_{CA} qui n'existent pas dans MOD_C et réciproquement. A priori, cette configuration est la plus proche de la réalité, car dans la réalité les modèles

du système ont toujours des parties communes sans qu'il n'y ait vraiment égalité des ensembles de comportements. La figure 5.3 illustre cette configuration.

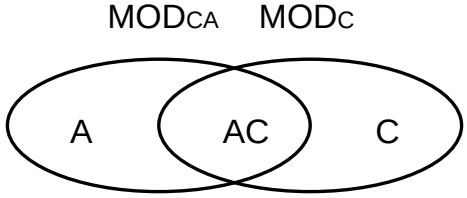


FIG. 5.3 – Intersection entre deux modèles

3. Il y a dans cette troisième configuration parfaite égalité entre les ensembles de comportements du modèle de commande et du modèle des comportements attendus. Bien que cette configuration paraisse idéale, elle met en lumière la redondance des deux modèles, un seul suffirait. Elle est montrée figure 5.4.

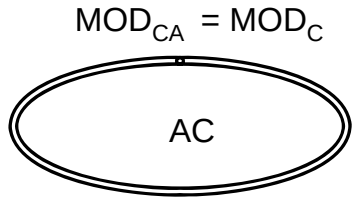


FIG. 5.4 – Deux modèles identiques

4. La quatrième configuration nous indique que tous les comportements décrits par le modèle de la commande MOD_C sont également des comportements du modèle du comportement attendu MOD_{CA}. Cela correspond à la configuration la plus logique dans laquelle la commande restreint les comportements possibles aux trajectoires de la loi de commande. Cette configuration correspond en particulier à la philosophie de la commande supervisée [RAMADGE et WONHAM, 1989]. La figure 5.5 nous montre ce cas.

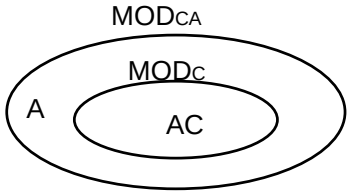


FIG. 5.5 – Commande restreignant les comportement attendus

5. Dans la cinquième configuration présentée dans la figure 5.6 nous retrouvons le cas dual du précédent. Ici ce sont les comportements attendus de MOD_{CA} qui constituent un sous ensemble des comportements de la commande. Dans le cadre de la commande de procédé manufacturiers flexibles, cette configuration correspond par exemple à un ordonnancement non entièrement défini (groupe d'opérations permutables par exemple) avec un choix fait au dernier moment en fonction de l'état courant d'occupation des ressources de production.

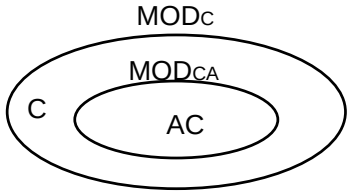


FIG. 5.6 – Commande englobant les comportement attendus

Si les configurations 1 et 3 sont peu réalistes, nous ne les considérons pas impossibles a priori car, même s'il est envisageable de détecter à l'avance ces configurations, nous nous plaçons dans nos travaux dans l'hypothèse où les modèles contiennent des erreurs et sont, de plus, modifiés en ligne. Il faut noter toutefois que la configuration 1 pourrait être totalement ignorée, car les modifications de modèles que nous proposerons dans le chapitre 6 ne peuvent jamais conduire à deux ensembles de comportements disjoints.

L'intérêt de connaître la configuration pour modifier les modèles est évident. Toutefois, nous allons montrer que l'obtention de cette connaissance n'est pas immédiate.

5.1.3 Principe de la détermination de la configuration en ligne

Pour une observation $T_R = T_{R1}.(M_{n-1}, e_{n-1}, M_n)$ avec, $(M_{n-1}, e_{n-1}, M_n) = T_{R2}$, trois situations sont possibles :

1. $T_{R2} \in TR(MOD_{CA}) \wedge T_{R2} \in TR(MOD_C)$. Ceci est le fonctionnement normal, nous l'avons déjà souligné, et traduit le fait que les deux modèles ne sont pas disjoints. Dans ce cas, la configuration 1 est éliminée car cette configuration ne correspond pas avec le cas évoqué ici.
2. $(T_{R2} \in TR(MOD_{CA}) \wedge T_{R2} \notin TR(MOD_C)) \vee (T_{R2} \notin TR(MOD_{CA}) \wedge T_{R2} \in TR(MOD_C))$. Ceci traduit la détection d'une incohérence et met en évidence que les modèles ne correspondent pas aux configurations 1 et 3. La détection de cette incohérence nous renseigne donc sur la configuration des ensembles de trajectoires l'un par rapport à l'autre. Mais il apparaît aussi qu'une seule observation va permettre d'éliminer deux configurations en conservant les trois autres.
3. $T_{R2} \notin TR(MOD_{CA}) \wedge T_{R2} \notin TR(MOD_C)$. Cette détection d'incohérence est possible quelle que soit la configuration des ensembles de trajectoires. En effet, il est toujours possible de voir apparaître une observation qui n'est représentée dans aucun modèle. Sauf à supposer, cas extrême, que l'un des modèles couvre l'intégralité des observations potentielles, cette observation est toujours possible et n'apporte aucune information.

Seule une succession d'observation, par exemple :

1. d'abord $T_{R2} \in TR(MOD_{CA}) \wedge T_{R2} \in TR(MOD_C)$ (élimination de la configuration 1),
2. ensuite $T_{R2} \in TR(MOD_{CA}) \wedge T_{R2} \notin TR(MOD_C)$ (élimination des configurations 3 et 5)
3. enfin $T_{R2} \notin TR(MOD_{CA}) \wedge T_{R2} \in TR(MOD_C)$ (élimination de la configuration 4)

permet d'acquérir la certitude sur la configuration des ensembles de trajectoires des modèles de bon fonctionnement. Il faut également remarquer que seule la configuration 2 peut être déterminée sans incertitude.

Cependant connaître l'ensemble de configurations possibles pour une observation en particulier, ne nous permet toujours pas de savoir s'il s'agit d'une erreur dans le modèle ou d'une défaillance. Pour cela nous allons utiliser un troisième modèle du système censé constituer un modèle idéal, nous allons supposer connaître ce modèle. Nous présentons dans la section suivante ce modèle et la façon dont il est utilisé.

5.2 Distinction des faux symptômes

Comme cela a été déjà mentionné, la détection d'une incohérence peut avoir trois origines :

1. une défaillance due à une faute d'un composant (symptôme de défaillance),
2. une erreur dans le modèle ne contenant pas de trajectoire cohérente avec l'observation (faux symptôme),
3. une erreur dans l'observation (défaillance d'un capteur).

Nous allons utiliser un troisième modèle du système pour distinguer un faux symptôme d'un symptôme de défaillance. Dans plusieurs travaux ce modèle est supposé être connu. Implicitement l'hypothèse que le modèle du système est correct et complet est faite. En réalité, il représente le modèle idéal de bon fonctionnement qui décrit très exactement tous les comportements possibles. Cette hypothèse n'est pas réaliste dans un système complexe, ce modèle étant déjà très difficile à construire pour des systèmes simples et déterministes comme, par exemple, des réalisations électroniques logiques. Nous allons lever en partie cette hypothèse en considérant que nous disposons d'un modèle de comportement réel que nous assimilons au modèle idéal.

5.2.1 Utilisation du modèle du comportement réel du système

Ce modèle est censé décrire de façon correcte et complète le fonctionnement normal du système. Nous le notons MOD_{CR}. En réalité, ce modèle est hypothétique et nous considérons que tant que MOD_{CR} peut être considéré sans erreur, il constitue ce modèle idéal. Il permet, comme dans toutes les approches de détection par rupture de cohérence, de déterminer si le procédé subit une évolution imprévue résultant d'une défaillance.

Ceci devient, bien sûr, possible quand on intègre le troisième modèle à $TR(MOD_{CA})$ et $TR(MOD_C)$ car comme le montre la figure 5.7, toute observation incluse dans $TR(MOD_{CR})$ représentera un comportement normal et réciproquement.

Si nous supposons que la dernière observation conduit à $T_{R2} \notin TR(MOD_{CA})$, en considérant l'appartenance de T_{R2} à $TR(MOD_{CR})$ nous pouvons déterminer s'il s'agit d'un faux symptôme ou d'un symptôme de défaillance. Considérons la figure 5.8. Dans cette figure le point noir en dehors de $TR(MOD_{CA})$ représente l'observation T_{R2} qui a été détectée incohérente avec MOD_{CA} lors de la phase de détection.

Comme T_{R2} appartient à $TR(MOD_{CR})$, on voit clairement que même si l'observation a été trouvée incohérente vis à vis de MOD_{CA}, il s'agit d'un comportement normal, T_{R2} appartenant à MOD_{CR}. Nous concluons naturellement qu'il s'agit d'un faux symptôme,

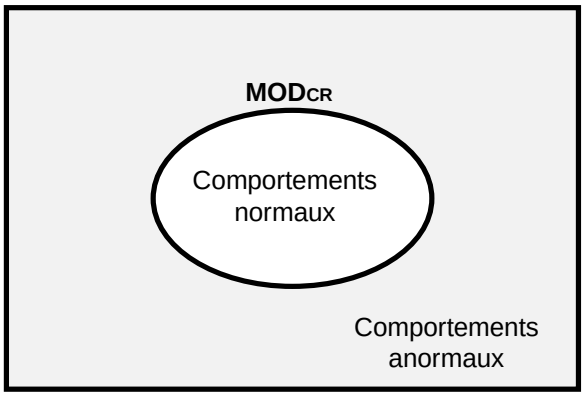


FIG. 5.7 – Le modèle de comportement réel du système, MOD_{CR}

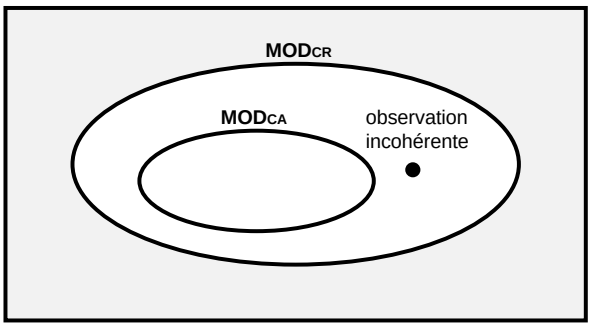


FIG. 5.8 – Observation d'une erreur dans le modèle MOD_{CA}

que la cause se trouve au niveau du MOD_{CA} et qu'en réalité il n'y a pas de défaillance. Mais, maintenant considérons que nos modèles soient agencés comme le montre la figure 5.9. Ici la conclusion sur l'incohérence montre qu'il s'agit d'une véritable défaillance dans le système car T_{R2} sort complètement des limites de MOD_{CR}.

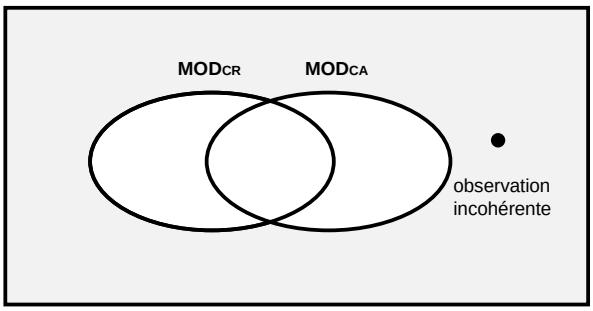


FIG. 5.9 – Observation en dehors de MOD_{CR} et MOD_{CA}

Comme nous ne connaissons pas réellement le modèle idéal, nous considérons que nous disposons d'une version approchée de MOD_{CR} dont nous ne remettrons en cause l'exactitude et la complétude qu'en tout dernier ressort, après avoir épuisé toutes les autres possibilités de rétablissement de la cohérence. Globalement, le système de surveillance fonctionne donc comme suit :

1-**Lors de la détection d'une incohérence** T_{R2}
2-**Si** $T_{R2} \in MOD_{CR}$
3- **Alors** diagnostic par modification des modèles MOD_C et/ou MOD_{CA}
4- **Sinon** diagnostic d'une défaillance d'un composant du procédé
5- **Si** (succès du diagnostic de défaillance)
6- **Alors** reconfigurer l'installation ou réparer
7- **Sinon** diagnostic par modification des modèles MOD_{CA}, MOD_C et MOD_{CR}

Cet algorithme ne fait pas apparaître les cas où la modification des observations (MOD_{OBS}) permet de restaurer la cohérence. L'intégration de ce mécanisme fait l'objet d'une perspective de cette thèse, mais n'a pas été encore abordée en détail.

Nous allons envisager toutes les configurations possibles des trois modèles : MOD_C, MOD_{CA} et MOD_{CR}. Ainsi, nous pourrons par observations successives déterminer la configuration réellement occupée par les modèles et par la suite conclure avec plus de précisions qu'en considérant, comme dans le cas classique, que toute incohérence traduit une défaillance ou, comme nous serions obligés de le faire sans ce troisième modèle, que toute incohérence est due à une erreur dans MOD_{CA} ou dans MOD_C.

5.2.2 Détermination des configurations entre MOD_{CA}, MOD_{CR} et MOD_C

Le problème est similaire par rapport à deux modèles, mais le nombre de cas est bien plus élevé. En effet, nous avons vu que le nombre de configurations pour k ensembles est de 2^k. Ici avec trois modèles, nous aurons 7 ensembles différents (figure 5.10). Avec les mêmes notations que précédemment et R pour représenter les trajectoires du modèles MOD_{CR}, nous obtenons :

$$
\begin{aligned}
A &: TR(MOD_{CA}) \backslash ((TR(MOD_{CA}) \cap TR(MOD_C)) \cup (TR(MOD_{CA}) \cap TR(MOD_{CR}))),\\
C &: TR(MOD_C) \backslash ((TR(MOD_C) \cap TR(MOD_{CA})) \cup (TR(MOD_C) \cap TR(MOD_{CR}))),\\
R &: TR(MOD_{CR}) \backslash ((TR(MOD_{CR}) \cap TR(MOD_{CA})) \cup (TR(MOD_{CR}) \cap TR(MOD_C))),\\
AC &: TR(MOD_{CA}) \cap TR(MOD_C) \backslash ((TR(MOD_{CA}) \cap TR(MOD_{CR}) \cap TR(MOD_C))),\\
AR &: TR(MOD_{CA}) \cap TR(MOD_{CR}) \backslash ((TR(MOD_{CA}) \cap TR(MOD_{CR}) \cap TR(MOD_C))),\\
CR &: TR(MOD_C) \cap TR(MOD_{CR}) \backslash ((TR(MOD_{CA}) \cap TR(MOD_{CR}) \cap TR(MOD_C))),\\
ACR &: TR(MOD_{CA}) \cap TR(MOD_{CR}) \cap TR(MOD_C),
\end{aligned}
$$

En fait avec 2 modèles nous avions $2^2 - 1$ ensembles, avec 3 modèles nous avons $2^3 - 1$ ensembles. La généralisation à n modèles n'est pas l'objet de cette étude. Elle pourrait être faite par récurrence. Elle peut également être déterminée par le raisonnement suivant. Pour une trajectoire et n modèles, il existe 2^n possibilités d'appartenance, de l'appartenance à aucun des modèles, à l'appartenance à tous. Dans les ensembles que nous considérons l'appartenance à aucun ensemble n'a pas de sens puisque nous recherchons les intersections des ensembles (il s'agirait de considérer l'intersection de ϕ avec les autres ensembles). En conclusion, nous avons bien $2^n - 1$ ensembles définis pour n modèles. Pour 3 modèles, nous obtenons donc 128 (2^7) configurations des ensembles de trajectoires des modèles.

Comme dans le cas à deux modèles, certaines configurations n'ont aucun intérêt. Il s'agit des configurations où l'on ne retrouve aucune des trajectoires d'au moins un modèle

(par exemple $\{A\}$ qui ne contient ni les trajectoires de MOD_{CR} ni celles de MOD_C). Les configurations éliminées sont 19 au total et elles sont répertoriées figure 5.11.

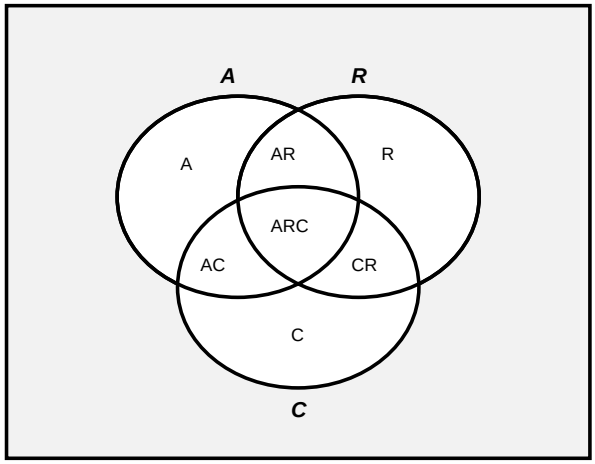


FIG. 5.10 – Les 7 ensembles pour trois modèles

Configuration formée par un ensemble	Configuration formée par deux ensembles	Configuration formée par trois ensembles
1.Ø	8. { A, R }	17. { A, R, AR }
2.{ A }	9. { A, C }	18. { A, C, AC }
3.{ R }	10. { A, AR }	19. { C, R, RC }
4.{ C }	11. { A, AC }	
5.{ AR }	12. { R, C }	
6.{ AC }	13. { R, AR }	
7.{ CR }	14. { R, RC }	
	15. { C, AC }	
	16. { C, RC }	

FIG. 5.11 – Élimination des configurations sans intérêt

Étant donné le nombre de configurations possibles pour les trois modèles, nous ne pouvons pas les représenter dans ce document. Nous allons seulement en montrer quelques unes au long de ce chapitre. L'intégralité de la liste est donnée en annexe page 165.

Présentons maintenant une méthode pour identifier au mieux la configuration de ces trois modèles.

5.3 La méthode d'identification de la configuration

Cette méthode est basée sur l'information engendrée par la phase de détection et les observations. Nous avons vu sur l'exemple à deux modèles qu'une succession d'observations pouvait améliorer la connaissance de la configuration des ensembles de trajectoires des différents modèles. Avec cette information nous allons éliminer certaines configurations. Nous conservons seulement la ou les configurations cohérentes à l'information

apportée par les observations. **L'identification fine de ces configurations est en effet vitale pour pouvoir faire réellement la différence entre les incohérences causées par une erreur du modèle et les vrais symptômes.** Pour cela, nous allons construire un graphe qui sera parcouru en ligne, à chaque observation, pour cerner de mieux en mieux la configuration des modèles. D'abord nous allons montrer comment on construit l'information à partir des observations.

5.3.1 Les informations sur les modèles

La seule connaissance certaine est que nous avons trois modèles : le modèle de comportement attendu, MOD_{CA} le modèle de comportement réel, MOD_{CR} et le modèle de commande, MOD_C. Ces modèles sont des éléments de l'univers Ω du problème d'identification,

$$\begin{cases} TR(MOD_{CA}) & \in & \Omega \\ TR(MOD_C) & \in & \Omega \\ TR(MOD_{CR}) & \in & \Omega \end{cases}$$

Nous rappelons que $TR(X)$ est l'ensemble fini des trajectoires finies décrites par le modèle X. Nous désignerons par TR_x une trajectoire appartenant à l'ensemble $TR(X)$.

5.3.2 Les informations déterminées à partir des observations

On sait qu'une séquence d'observation S_{obs} peut être cohérente ou incohérente avec chacun des modèles. Ainsi, chaque observation peut être classée dans une des huit possibilités de cohérence ou d'incohérence par rapport aux modèles (figure 5.12).

N° possibilité	MOD_{CA}	MOD_{CR}	MOD_C
1.-	Incohérente	Incohérente	Incohérente
2.-	Incohérente	Incohérente	Cohérente
3.-	Incohérente	Cohérente	Incohérente
4.-	Cohérente	Incohérente	Incohérente
5.-	Incohérente	Cohérente	Cohérente
6.-	Cohérente	Incohérente	Cohérente
7.-	Cohérente	Cohérente	Incohérente
8.-	Cohérente	Cohérente	Cohérente

FIG. 5.12 – Les 8 possibilités d'incohérence pour une observation

Quand une séquence d'observation S_{obs} est cohérente avec un modèle MOD_i, elle nous donne de l'information sur l'existence dans MOD_i d'une trajectoire dont la partie observable, les événements, est justement S_{obs}. Plus précisément, une observation o_i est cohérente avec MOD_i quand il existe une transition t_i associée à la dernière observation o_i ($Ev(t_i) = Eo(o_i)$) franchissable à partir de l'état courant de MOD_i. On en déduit que le comportement observé est aussi décrit dans le modèle en question. Nous faisons en effet l'hypothèse que le procédé ne génère jamais, lors d'une défaillance, une séquence correspondant à une trajectoire comportant une erreur de modélisation.

Si, au contraire, une observation est incohérente avec le modèle MOD_i, alors l'événement associé à l'observation ne correspond à aucune transition franchissable à partir de l'état courant de MOD_i. Ce qui implique que le comportement observé n'est pas décrit dans MOD_i.

Ce type d'informations peut être utilisé pour réduire l'ensemble de configurations possibles, en gardant seulement les configurations cohérentes avec les observations. Illustrons ceci par un exemple. Si nous prenons la possibilité décrite à la ligne 5 dans le tableau de la figure 5.12. Elle nous indique que l'observation reçue est incohérente par rapport aux comportements de MOD_{CA}, cohérente avec les comportements de MOD_{CR} et de MOD_C. Avec cette information, il est possible de déterminer toutes les configurations dans lesquelles cette situation est possible. Par exemple, dans la figure 5.13, représentant deux des cent neuf configurations possibles pour trois modèles, la configuration de figure 5.13.a) permet de représenter l'appartenance de l'observation à MOD_{CR} et de MOD_C. Dans cette figure, l'observation correspond au point noir en dehors de MOD_{CA} pour représenter la incohérence avec ce modèle, mais dans MOD_C et MOD_{CR} pour indiquer la cohérence avec ces deux modèles.

En revanche, cette observation ne peut pas être représentée dans la configuration de la figure 5.13.b) car il n'existe pas une région correspondant à la constatation (incohérente avec MOD_{CA}, cohérente avec MOD_{CR} et cohérente avec MOD_C). Il est donc possible d'affirmer que les modèles ne sont pas agencés comme le montre la figure 5.13.b.

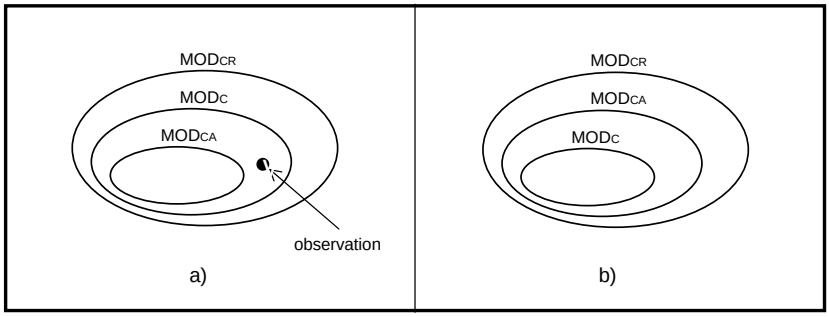


FIG. 5.13 – (a)Placement de l'observation, (b)impossibilité de placement de l'observation

En procédant comme nous l'avons mentionné, nous pouvons éliminer toutes les configurations qui ne correspondent pas à la constatation réalisée par l'observation reçue. Ainsi, nous obtenons un ensemble de configurations réduit où il est possible de représenter cette observation. Formalisons ces concepts afin de décrire le mécanisme que nous venons d'évoquer.

Soit CNS une constatation, c'est à dire une caractérisation de la cohérence d'une observation par rapport aux trois modèles. Notons (CNS_Y), dans cette constatation, la cohérence de l'observation en regard du modèle Y. Cette constatation est une fonction booléenne définie par,

$$\Omega \xrightarrow{CNS_Y} B = \{0, 1\}$$

avec

$$\begin{cases} CNS_Y(TR_x) & = 0, \quad TR_x \notin Y \\ CNS_Y(TR_x) & = 1, \quad TR_x \in Y \end{cases}$$

La constatation CNS effectuée situant cette observation TR_x par rapport aux trois modèles A, C et R, est définie par une fonction Booléenne vectorielle

$$\Omega \xrightarrow{CNS_{ARC}} B^3$$

avec

$$CNS_{ARC}(TR_x) = (CNS_A(TR_x), CNS_R(TR_x), CNS_C(TR_x))$$

Les 8 constatations possibles représentées par les valeurs de $CNS_{ARC}(TR_x)$ sont données dans la figure 5.14, autre écriture du tableau de la figure 5.12.

Revenons encore une fois sur le cas de la constatation numéro 1 de la figure 5.14, $CNS_{ARC}(TR_x) = (0, 0, 0)$. Elle indique que l'observation représentée par TR_x, est incohérente avec les trois modèles. Cependant, même si dans ce cas TR_x n'appartient à aucun de trois modèles, $TR_x \notin TR(MOD_{CA})$, $TR_x \notin TR(MOD_{CR})$, $TR_x \notin TR(MOD_C)$, elle appartient tout de même à Ω (comme l'illustre la figure 5.15). Alors cette constatation nous indique que $A \cup R \cup C \neq \Omega$. Le cas contraire ne serait pas vrai, c'est-à-dire, si on constate que toutes les TR_x observées sont cohérentes avec au moins l'un des modèles, cela n'implique pas que $A \cup R \cup C = \Omega$. En effet, ne pas observer un élément d'un ensemble ne nous permet pas de conclure que cet ensemble n'existe pas.

N° CNS_{ARC}	A	R	C
1.-	0	0	0
2.-	0	0	1
3.-	0	1	0
4.-	1	0	0
5.-	0	1	1
6.-	1	0	1
7.-	1	1	0
8.-	1	1	1

FIG. 5.14 – Les valeurs de la constatation pour les trois modèles

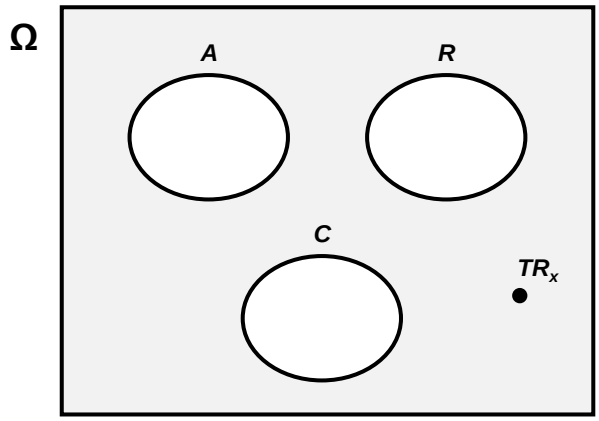


FIG. 5.15 – Appartenance de TR_x à Ω

Chaque constatation $CNS_{ARC}(TR_x)$, nous apporte des informations importantes pour l'identification de la configuration. Par exemple, la valeur de la constatation $CNS_{ARC}(TR_x) = (1, 1, 1)$ indique que l'observation est cohérente avec les trois modèles. On peut en déduire que $TR_x \in TR(MOD_{CA})$, $TR_x \in TR(MOD_R)$, $TR_x \in TR(MOD_C)$ et donc que l'ensemble noté ACR existe dans la configuration des modèles. La figure 5.10 est une configuration qui contient l'ensemble ARC. Une prochaine observation dont la valeur de la constatation est différente des valeurs déjà observées permettra de faire évoluer la connaissance au sujet de la configuration des ensembles. Par exemple, une observation dont la valeur de la constatation est $CNS_{ARC}(TR_x) = (1, 0, 0)$, implique l'existence de l'ensemble noté A. Cette séquence de constatations nous apporte une certaine connaissance, par exemple nous connaissons maintenant que notre configuration contient les deux ensembles ARC et A.

Notons TC une séquence de constatations $CNS_{ARC}(TR_x)$ obtenue de la séquence d'observations S_{obs}. Cette séquence de constatations TC est définie par,

$$TC = \langle CNS_{ARC}(TR_1), CNS_{ARC}(TR_2), ...CNS_{ARC}(TR_n)\rangle$$

Les observations TR_i qui produisent une séquence de constatations TC forment son ensemble support noté $S(TC)$ défini par,

$$S(TC) = \{TR_1, TR_2, ...TR_n\}$$

où

$$S(TC) \subset \Omega$$

Par exemple supposons que la séquence d'événements observés soit $S_{obs} = e_2e_3e_5$ et que la séquence de constatations produits par S_{obs} soit $TC_1 = \langle(0, 0, 1), (0, 0, 1), (0, 0, 0)\rangle$, alors l'ensemble support de TC_1 est $S(TC_1) = \{e_2, e_3, e_5\}$.

La connaissance associée à une séquence de constatations est notée $K_{ARC}(TC)$, cette connaissance est définie par,

$$K_{ARC}(TC) = \bigcup_{i=1}^{n} \{CNS_{ARC}(TR_i \mid TR_i \in TC)\}$$

Dans l'exemple ci-dessus la connaissance vaut

$$K_{ARC}(TC) = \{(0, 0, 1), (0, 0, 0)\}$$

Notons que ni l'ordre des constatations, ni la répétition d'une constatation, n'a d'influence sur la valeur de la connaissance. Seule des constatations différentes de celles déjà obtenues par les observations précédentes font évoluer la connaissance.

Démonstrations :

- L'ordre n'a pas d'influence sur la valeur de la connaissance.

Soient deux séquences distinctes construites sur le même ensemble support,

$$\forall i, j = [1, .., n], CNS_{ARC}(TR_i) = CNS_{ARC}(TR_j) \Longleftrightarrow i = j$$

Soient deux séquences de constatations

$$TC_1 = \langle \ldots, CNS_{ARC}(TR_l), CNS_{ARC}(TR_{l+1}), \ldots\rangle$$

et

$$TC_2 = \langle \ldots, CNS_{ARC}(TR_{l+1}), CNS_{ARC}(TR_l), \ldots \rangle$$

ne différant que par l'ordre des constatations TR_l et TR_{l+1}. On a

$$K_{ARC}(TC_1) = \{..., CNS_{ARC}(TR_l), CNS_{ARC}(TR_{l+1}), \ldots\}$$

et

$$K_{ARC}(TC_2) = \{..., CNS_{ARC}(TR_{l+1}), CNS_{ARC}(TR_l), \ldots\}$$

qui montre bien que

$$K_{ARC}(TC_1) = K_{ARC}(TC_2)$$

- La répétition d'une constatation ne modifie pas la connaissance.
Il suffit de montrer que

$$TC_1 = \langle CNS_{ARC}(TR_l) \rangle$$

et

$$TC_2 = \langle CNS_{ARC}(TR_l), CNS_{ARC}(TR_l) \rangle$$

produisent la connaissance, ce qui est immédiat par application de la définition.

Corollaire : réduction de l'ensemble support d'une séquence de constatations $S(TC)$
Si deux éléments TR_j, TR_k de l'ensemble support $S(TC)$, sont tels que

$$(TR_j \neq TR_k) \wedge ((CNS_{ARC}(TR_j) = CNS_{ARC}(TR_k)),$$

alors l'ensemble support $S(TC_r) \setminus \{TR_k\}$ va générer des séquences de constatations dont la connaissance est identique aux séquences de constatations que peut générer l'ensemble support initial $S(TC)$.

La démonstration de cette propriété est immédiate par application de la définition de la connaissance aux deux ensembles supports du corollaire.

L'exemple précédent illustre ceci puisque les éléments e_2 et e_3 sont différents alors qu'ils engendrent la même constatation $(0, 0, 1)$.

Un ensemble support possible dans cet exemple aurait donc pu être : $\{e_2, e_5\}$

Avant de montrer comment cette connaissance est utilisée en ligne pour identifier la configuration des modèles, montrons d'abord comment le graphe des configurations est établi hors ligne.

5.3.3 Graphe d'identification des configurations

Ce graphe va nous permettre d'utiliser les informations contenues dans les constatations définies précédemment. Cependant, pour une constatation donnée la connaissance n'est jamais suffisante pour déterminer avec certitude la configuration dans laquelle se trouvent les modèles. Chaque nouvelle constatation apportant une connaissance nouvelle fait progresser la connaissance globale sur la configuration. Le graphe que nous proposons permet de représenter facilement ce mécanisme de cumul de la connaissance. Il est construit hors ligne puisqu'il recouvre tous les cas possibles, et c'est en suivant les arcs

correspondant aux constatations dans l'état courant de connaissance que nous déterminons, sans calcul supplémentaire en ligne, le nouvel état de connaissance. Comme dans le cas des deux modèles, une seule configuration permet de lever toutes les incertitudes.

Cette configuration est montrée dans la figure 5.10 pour les cas de trois modèles. Elle correspond au cumul des sept constatations différentes possibles.

Construction du graphe

Rappelons que pour 3 modèles, apparaissent au plus 7 "régions" appelées ensembles, $\langle A, C, R, AC, AR, CR, ARC\rangle$ définissant 109 configurations possibles. Avant toute constatation, dans l'état initial de ce graphe, ces 109 configurations sont possibles. Il possède donc un sommet initial. Ce graphe va permettre de cumuler des connaissances, il est donc orienté, son parcours correspond à l'augmentation de la connaissance globale de la configuration des modèles. Il est caractérisé comme suit.

- Les sommets du graphe sont étiquetés par le n-uplet des ensembles dont l'existence a pu être déterminée avec certitude par parcours du graphe depuis son sommet initial. Bien sûr, pour un n-uplet donné plusieurs configurations sont généralement admissibles. Par exemple, à un sommet dont l'étiquette est constituée par l'ensemble $\{A, C\}$ sont associées toutes les configurations des trois modèles dans laquelle A et C apparaissent.
- Les arcs du graphe sont étiquetés avec les constatations $CNS_{ARC}(TR_x)$ réalisées.

A partir du sommet initial 7 constatations sont possibles (la constatation $(0, 0, 0)$ n'apportant aucune information). Les interprétations des 7 arcs sont présentées dans la figure 5.16.

Valeur de la constatation possible	A	R	C	AR	AC	RC	ARC
1.-(0, 0, 1)			X				
2.-(0, 1, 0)		X					
3.-(0, 1, 1)						X	
4.-(1, 0, 0)	X						
5.-(1, 0, 1)					X		
6.-(1, 1, 0)				X			
7.-(1, 1, 1)							X

FIG. 5.16 – Interprétation des arcs du graphe d'identification

Les niveaux du graphe

Comme nous l'avons introduit précédemment, ce graphe est orienté, c'est à dire qu'il ne possède pas de boucle, et sa profondeur, c'est à dire la longueur maximale d'une trajectoire est de 7 constatations. A chaque constatation apportant un nouvel élément de connaissance, on progresse d'un niveau dans le graphe. Toutes les trajectoires convergent vers le nœud terminal associé à l'étiquette $\{A, C, R, AC, AR, CR, ARC\}$. Les sept arcs constituant la trajectoire suivie apportant chacun un des ensembles comme élément de connaissance.

Détaillons niveau par niveau la constitution de ce graphe qui ne peut être représenté en raison de sa taille.

Niveau 0 : le niveau 0 est constitué par le sommet initial. 7 arcs sont issus de ce sommet initial. Ils correspondent aux 7 constatations possibles. En fonction de la connaissance apportée par la constatation, l'arc est relié aux sommets du niveau 1.

Niveau 1 : 7 sommets différents correspondant aux 7 constatations différentes possibles à partir du sommet initial constituent ce niveau. A chacun de ces sommets est déjà associée une constatation et la connaissance qui lui est liée. Ne sont donc prises en compte que les constatations ayant une connaissance associée différente de celle du sommet considéré, soit six au total à partir de chaque sommet.

Niveau 2 : d'après ce que nous venons de voir au sujet du niveau 1, le niveau 2 semble constitué de 7×6 sommets. En fait nous avons montré précédemment que l'ordre des constatations n'intervient pas dans la connaissance qu'apporte une séquence de constatation, il n'y a donc que $(7 \times 6)/2 = 21$ sommets au niveau 2. Pour les mêmes raisons que précédemment, 5 arcs partent de chaque sommet correspondant aux cinq constatations non encore perçues au sommet considéré.

Niveau 3 : le niveau 3 semble constitué par $21 \times 5 = 110$ sommets, mais comme précédemment, il apparaît que tous ces sommets n'ont pas à être distingués puisqu'ils n'apportent pas tous une connaissance différente. En fait, le nombre total de sommets du niveau 3 est plutôt donné par $7 \times 6 \times 5$ (nombre de trajectoires différentes menant du niveau 1 au niveau 3, divisé par $3 \times 2 \times 1$, le nombre de séquence de constatation du niveau trois apportant la même connaissance, c'est à dire construite sur le même support.

Généralisation

A un niveau n, $1 \leq n \leq 7$ le nombre de sommets est donné par le nombre de combinaisons de n constatations parmi les 7 possibles.

- Au niveau 0 : état initial 1
- Au niveau 1 : $\mathcal{C}_7^1$ =7 sommets,
- Au niveau 2 : $\mathcal{C}_7^2$ =21 sommets,
- Au niveau 3 : $\mathcal{C}_7^3$ =35 sommets,
- Au niveau 4 : $\mathcal{C}_7^4$ =35 sommets,
- Au niveau 5 : $\mathcal{C}_7^5$ =21 sommets,
- Au niveau 6 : $\mathcal{C}_7^6$ =7 sommets,
- Au niveau 7 : $\mathcal{C}_7^7$ =1 sommets,

Nous avons donc un graphe comportant 128 sommets qui correspondent bien à tous les états de la connaissance au regard des 7 ensembles dont nous avons à déterminer l'existence.

Notons tout de même que si le nombre de sommets de ce graphe n'est pas très élevé, le nombre de séquences de constatations qu'il décrit est très élevé. Il y a ici toute la combinatoire d'enchaînement de 7 constatations différentes parmi 7 possibles qui vaut $7! = 5040$.

C'est la raison pour laquelle seule une représentation partielle de ce graphe est montrée dans la figure 5.17. Notons que, pour des raisons de lisibilité, chaque sommet est étiqueté uniquement par la connaissance apportée par la dernière constatation. La connaissance d'un sommet correspond à l'union des étiquettes de la branche qui lui est associée.

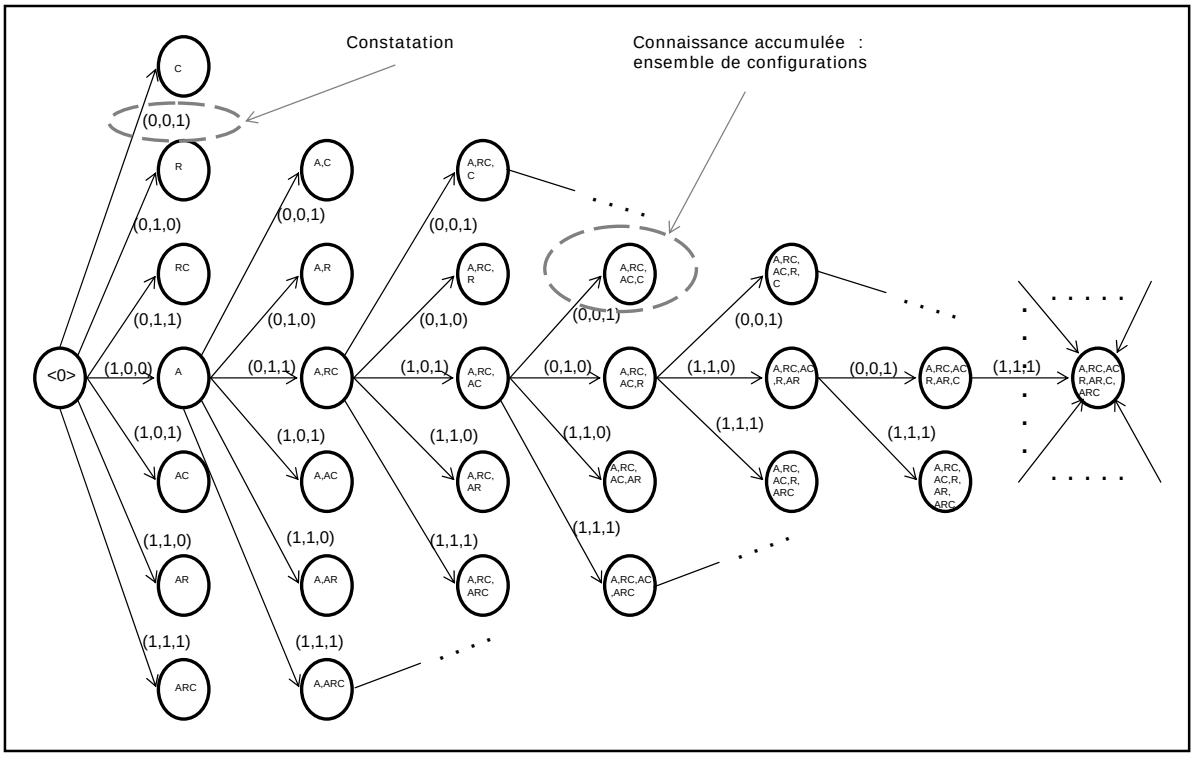


FIG. 5.17 – Représentation partielle du graphe d'identification de la configuration

Un programme en C a été implémenté pour la construction de la totalité de ce graphe. La structure de données qu'il génère est prête pour une utilisation en ligne dans le but d'identifier la configuration.

Ce graphe est parcouru en ligne à chaque constatation. Comme nous l'avons mentionné auparavant, 7 constatations différentes sont nécessaires pour lever toute incertitude et aboutir à la configuration déjà évoquée dans laquelle les 7 ensembles sont présents.

Toutes les trajectoires convergent vers cette configuration, mais elles passent par des états intermédiaires dans lesquels, même s'il reste une incertitude (plus d'une configuration est admissible), l'identification de la configuration est déjà effectuée en partie. Il faut noter que le graphe n'est pas forcément parcouru jusqu'à son sommet final. En effet, si la configuration des modèles n'est pas celle du sommet final, jamais la séquence de 7 constatations différentes ne sera perçue.

5.4 Application de l'identification de la configuration au diagnostic

Une fois identifiée la (ou les) configuration(s) des modèles, le diagnostic devient possible. Toutefois, dans le cas où la solution est constituée de plus d'une configuration, il va falloir considérer chacune des configurations pour effectuer le diagnostic. Dans notre approche le diagnostic correspond à une modification du (ou des) modèle(s) pour rétablir la cohérence. Cette partie du diagnostic est complètement développée dans le chapitre 6 de ce manuscrit.

Ainsi le but de l'étape d'identification de la configuration est de doter à la fonction diagnostic de l'ensemble des configurations possibles, pour pouvoir dans l'étape de diagnostic déterminer s'il s'agit d'une erreur dans le modèle ou d'un véritable symptôme de défaillance.

Conclusions

Dans ce chapitre, nous décrivons la phase postérieure à l'étape de la détection de notre approche de diagnostic à base de cohérence.

Pour une observation donnée tous les cas d'incohérence ont été examinés. Nous avons souligné dans ce chapitre que ces erreurs dans les modèles provoquent les mêmes incohérences que les défaillances du procédé. Nous avons proposé dans ce chapitre, comme cela est fait dans toutes les approches, de nous appuyer sur un autre modèle appelé modèle de comportement réel noté MOD_{CR} supposé décrire la réalité d'une façon idéale sans erreurs d'aucun type.

Nous avons également proposé une démarche pour déterminer la configuration des ces trois modèles, à partir des observations reçues. Cette technique utilise un graphe construit hors ligne et qui est parcouru en ligne à la réception des observations issues du fonctionnement du système. Le parcours du graphe nous permet d'identifier un ensemble solution contenant uniquement les configurations cohérentes avec les observations récupérées sur le système. C'est cet ensemble-solution qui sera le point de départ de la fonction diagnostic, objet du chapitre suivant.

Chapitre 6

La fonction Diagnostic

Introduction

Ce chapitre est consacré au détail de la méthode du diagnostic de notre approche générale de détection et diagnostic basée cohérence. L'étape de diagnostic suit toujours l'étape de détection d'une incohérence sans laquelle il n'y a rien à diagnostiquer. Dans le chapitre 3 nous avons déjà présenté le but de la méthode de diagnostic qui est de rétablir la cohérence entre la réalité et la référence. Ce rétablissement de cohérence est possible grâce à la modification de la référence i.e. des modèles du système.

Les modifications réalisées sur les modèles par le diagnostic doivent respecter certaines propriétés des modèles. Nous allons donc commencer ce chapitre par la présentation des propriétés considérées au niveau du système et représentées dans son modèle. Nous aborderons ensuite les différentes techniques de modification des modèles notamment celles modifiant les coefficients de la matrice d'incidence des réseaux de Petri.

6.1 Propriétés du système

Une propriété d'un système représente des caractéristiques qui doivent être conservées ou vérifiées par le fonctionnement normal du système. De façon générale, deux types de propriétés peuvent être distingués : les propriétés structurelles et les propriétés comportementales. Ces propriétés sont définies dans le cahier des charges du système et, dans le cas d'une modélisation correcte, doivent être retranscrites dans les modèles du système.

Si nous considérons la possibilité d'erreurs dans le modèle, il est possible que certaines des propriétés du système soient omises ou modifiées, c'est à dire mal représentées.

L'idée de base du diagnostic est que les modifications apportées aux modèles doivent respecter les propriétés présentes dans le modèle original dans la mesure où ces propriétés n'empêchent pas de reproduire le comportement observé du système. Dans le cas contraire, il est possible, qu'après modification, certaines propriétés ne soient pas conservées ou que de nouvelles propriétés soient introduites. Dans tous les cas, seule la partie du modèle ne correspondant pas au comportement observé est affectée par ces modifications.

Les principales propriétés que nous considérons pertinentes sont celles associées à des caractéristiques structurelles comme par exemple dans le cadre des systèmes de

production : la quantité de circuits élémentaires de production, la quantité de ressources circulant dans chaque sous-processus, etc.

Nous considérons également importantes les propriétés comportementales de réinitialisation et de vivacité d'un système. En effet, un système doit pouvoir fonctionner en continu, sans arrêts intempestifs dans son fonctionnement. Autrement dit, les cycles de production doivent pouvoir s'enchaîner et se répéter ; un arrêt soudain peut alors être provoqué par une défaillance du système.

Au niveau d'un réseau de Petri (*RdP*), les propriétés comportementales dépendent du marquage initial et de la structure du RdP. Trois propriétés comportementales de base peuvent être recherchées au niveau d'un RdP.

- La vivacité,
- La bornitude,
- La réinitialisabilité.

Les propriétés structurelles dépendent principalement de la structure du réseau (liens entre places et transitions) et sont traduites par l'existence d'invariants de places et/ou de transitions au niveau du réseau de Petri.

Un invariant de places est une fonction linéaire des marquages dont la valeur constante dépend du marquage initial. L'ensemble Ps_r des places mises en jeu dans cette relation est donné par :

$$Ps_r = \{p \in P \backslash V_{Pr}(p) > 0\} \text{ avec } P \text{ l'ensemble des places du réseau de Petri et } V_{Pr} \backslash (V_{Pr})^T \bullet C = 0$$

V_{Pr} est le support de l'invariant de places r.

L'ensemble des invariants de places est Ps avec

$$Ps = \bigcup_1^m Ps_r \text{ avec } m \text{ le nombre d'invariants de places}$$

La recherche des invariants de places peut se faire par triangularisation de la matrice d'incidence C selon la méthode de Gauss.

Un invariant de transitions est donné par une séquence de tir σ_r qui ne modifie par le marquage du réseau. L'ensemble Ts_r des transitions mises en jeu dans l'invariant est donné par :

$$Ts_r = \{t \in T \backslash V_{Tr}(t) > 0\} \text{ avec } T \text{ l'ensemble des transitions du réseau de Petri et } V_{Tr} \backslash C \bullet V_{Tr} = 0$$

V_{Tr} est le support de l'invariant de transition r c'est à dire le vecteur caractéristique de la séquence σ_r.

L'ensemble des invariants de transitions est Ts avec

$$Ts = \bigcup_1^k Ts_r \text{ avec } k \text{ le nombre d'invariants de transitions}$$

La recherche des invariants de transitions peut se faire selon une méthode similaire à celle utilisée pour les invariants de places.

6.2 Le principe du diagnostic

Dans la littérature existante au niveau du diagnostic, après l'étape de détection d'un symptôme de défaillance, l'étape de diagnostic est chargée de localiser les composants défaillants, d'identifier les causes et de donner les explications nécessaires sur les défaillances.

Notre approche du diagnostic se distingue de cette vision classique du diagnostic dans la mesure où nous considérons possibles les erreurs de modélisation. Les incohérences mises en évidence par la détection peuvent donc aussi bien traduire de réelles défaillances au niveau du système surveillé (vrai symptôme), que des erreurs de modélisation dans les modèles du système pris comme référence du bon comportement (faux symptôme). Ceci implique que l'objet de notre approche de diagnostic n'est pas simplement le système au travers de ses composants mais également le modèle du système. En ce sens notre approche se distingue de la plupart des approches existantes.

Nous nous plaçons dans le cas où le diagnostic va supprimer l'incohérence en modifiant le modèle du système de manière à ce que ce dernier puisse à nouveau représenter le comportement observé.

La modification réalisée par le diagnostic va porter sur le ou les modèles mis en cause lors de la détection :

- si une observation est incohérente avec MOD_{CA} et cohérente avec MOD_C, nous devons modifier MOD_{CA} pour représenter le comportement observé et ainsi rétablir la cohérence entre MOD_{CA} et les observations.
- si une observation est cohérente avec MOD_{CA} mais incohérente avec MOD_C, MOD_C devra être modifié pour être à nouveau cohérent avec le comportement observé. Rappelons que même si MOD_C représente le modèle de la commande, nous ne le considérons pas exempt d'erreurs.
- si l'observation est incohérente avec les deux modèles MOD_{CA} et MOD_C, les deux modèles devront être modifiés pour absorber le comportement observé.

La modification du ou des modèles incohérents avec une observation est réalisée dès qu'un symptôme est détecté sans savoir a priori s'il s'agit d'une incohérence due à une erreur de modélisation ou d'un vrai symptôme de défaillance, et ceci dans le but de rétablir la cohérence et de caractériser les modifications appliquées aux modèles.

Comme nous l'avons déjà dit (chapitre 3), ces modifications sont réalisées avec le souci, d'une part, de respecter les propriétés initialement présentes dans le modèle à condition que celles-ci soient compatibles avec le comportement conservé et, d'autre part, de conserver la représentation des comportements observés du procédé qui ont été trouvés cohérents auparavant.

6.3 Les types de modifications envisagés

Sur la base de l'utilisation de modèles réseaux de Petri nous proposons trois types de modifications permettant de rétablir la cohérence entre les modèles et les observations. Dans la mesure où au niveau d'un réseau de Petri l'incohérence se manifeste par l'impos-

sibilité de franchir la transition associée à l'événement observé, les types de modifications proposés ont toutes pour objectif de rendre franchissable la transition incriminée.

1. **Modifications des coefficients de la matrice d'incidence** C**.** Il s'agit de modifier les poids des arcs i.e les coefficients de la matrice d'incidence C, sans modifier la dimension de la matrice (nombre de lignes et de colonnes) (figure 6.1). Cette modification est réalisée plus spécifiquement sur la matrice d'incidence avant (Pre) car, dans le réseau de Petri, une incohérence se traduit par le non franchissement de la transition associée à l'événement observé, cette transition n'étant pas sensibilisée. Pour pouvoir franchir cette transition une solution consiste à changer la (ou les) place(s) d'entrée de cette transition par des places marquées pour le marquage courant du réseau.

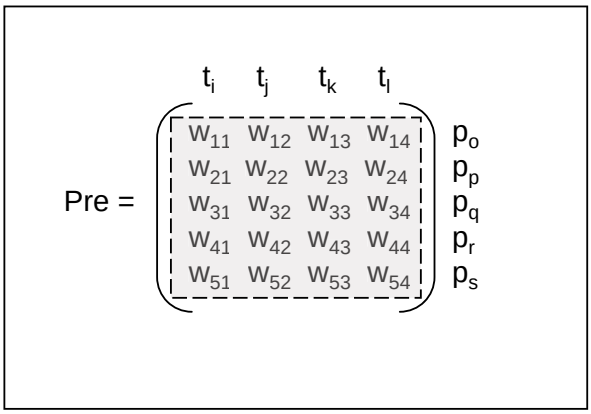


FIG. 6.1 – Modification des coefficients de la matrice d'incidence Pre

2. **Modification de la dimension de la matrice d'incidence C.** Il s'agit de modifier le nombre de lignes et/ou de colonnes de la matrice d'incidence en les augmentant ou en les diminuant, sans changer les coefficients de la matrice (voir figure 6.2). Cette modification consiste donc à rajouter de nouvelles places et de nouvelles transitions au niveau du réseau de Petri de telle sorte que le modèle modifié puisse représenter le comportement observé.

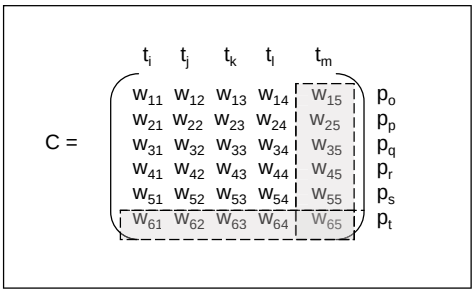


FIG. 6.2 – Modification de la structure de la matrice d'incidence C

3. **Modification du marquage du réseau.** Le but est de modifier uniquement le marquage du réseau de Petri, afin de sensibiliser la transition associée à l'événement observé, et reproduire ainsi le comportement observé trouvé incohérent.

Les matrices modifiées sont appelées matrices de Rétablissement de Cohérence C_{RC}. La mise en place de ces matrices a pour conséquence d'autoriser de nouveaux comportements au niveau des modèles et donc de créer un nouvel espace d'état cohérent avec celui du modèle d'observations. Le modèle du système va être ainsi élargi. Il est possible dans ces conditions que de nouvelles propriétés apparaissent dans les modèles modifiés.

Dans le cadre de cette thèse, nous nous sommes intéressés seulement à la modification des coefficients de la matrice d'incidence pour rétablir la cohérence. La section suivante donne le détail de la technique développée.

6.4 La méthode de modification des coefficients

Dans le modèle réseau de Petri, une incohérence se traduit par le non franchissement de la transition associée à l'événement observé, cette transition n'étant pas sensibilisée.

En changeant la ou les place(s) d'entrée(s) de cette transition par des places marquées pour le marquage courant du réseau, il est possible de rétablir la sensibilisation de la transition. Modifier les places d'entrée d'une transition revient à modifier la matrice d'incidence avant du réseau de Petri (Pre).

Les modifications doivent, rappelons-le, respecter les propriétés structurelles du système, qui au niveau d'un modèle RdP sont données par les invariants linéaires de places et de transitions.

Dans notre cas, les invariants de transitions dans lesquels la transition à l'origine de la détection d'incohérence entre en jeu, sont modifiés. En effet, seules sont conservées les propriétés compatibles avec le comportement conservé.

L'algorithme de modification de modèle est le suivant :

1. Déterminer pour le marquage courant M_k du RdP du modèle à modifier (MOD_{CA} ou MOD_C), l'ensemble des places marquées PM_k (supposé non vide).
2. Déterminer l'ensemble des nouvelles places d'entrée potentielles de la transition à l'origine de la détection d'incohérence t_{inc}. Cet ensemble noté M_{poss} est l'ensemble des modifications possibles.

   $$M_{poss} = \{Ep_i | i = 1..2^q\} \text{ avec } q \text{ le nombre de places contenues dans } PM_k$$

3. Construire l'ensemble Pre_{poss} des vecteurs des places d'entrée à t_{inc} possibles.

   $$Pre_{poss} = \{Pre_i(., t_{inc}) \backslash Pre_i(., t_{inc}) = [x_1, .., x_n]^T\}$$

   avec n le nombre de places du réseau de Petri à modifier
   et $Pre_i(., t_{inc})$ la colonne de la matrice Pre associée à la transition t_{inc} et au sous-ensemble de places Ep_i

   Pre_i est calculé de la façon suivante :

   $$x_j = 1 \text{ si } p \in Ep_i$$
   $$x_j = 0 \text{ si } p \notin Ep_i$$

4. Construire l'ensemble C_{poss} des vecteurs d'incidence associés à la transition t_{inc}.

   $C_{poss} = \{C_i(., t_{inc}) \mid C_i(., t_{inc}) = Post(., t_{inc}) - Pre_i(., t_{inc})\}$ avec $Post(., t_{inc})$ la colonne de la matrice $Post$ associée à la transition t_{inc} et $Pre_i(., t_{inc}) \in Pre_{poss}$

5. Déterminer l'ensemble EI_r des éléments de C_{poss} qui conservent l'invariant de places du RdP original donné par le support V_{Pr}. L'ensemble EI_r est donné par :
$$EI_r = \{C_i(., t_{inc}) | V_{Pr}^T . C_i(., t_{inc}) = 0\}$$
6. Déterminer l'ensemble E_{sol} des éléments de C_{poss} qui conservent l'ensemble de tous les invariants de places du RdP initial :
$$E_{sol} = \{C_i(., t_{inc}) | C_i(., t_{inc}) = (EI_1 \cap EI_2 \cap EI_2 \ldots \cap EI_m)\}$$
Où m représente le nombre d'invariants de places du RdP original. La solution $E_{sol} = \emptyset$, indique qu'il n'existe pas de modification conservant tous les invariants de places. Une solution contenant plus d'un élément dans l'ensemble E_{sol}, indique qu'il existe plus d'une modification conservant les invariants de places originaux.
7. Échanger l'ancien vecteur colonne de la matrice d'incidence associé à la transition à l'origine de l'incohérence par une des solutions trouvées, c'est à dire par un élément de E_{sol} :
$$C(., t_{inc}) = C_i(., t_{inc}) \quad \forall C_i(., t_{inc}) \in E_{sol}$$
La matrice d'incidence obtenue est la matrice de Rétablissement de Cohérence notée C_{RC}. Il y a autant de matrices C_{RC} que d'éléments dans E_{sol}.
8. Construire le nouveau modèle RdP du système défini par une des matrices de Rétablissement de Cohérence.

L'application de cet algorithme permet de conserver au niveau du nouveau modèle les invariants de places et de transitions qui ne sont pas liés à la transition à l'origine de l'incohérence.

Afin de mieux expliquer notre approche de diagnostic nous allons illustrer l'algorithme précédent sur un exemple choisi volontairement simple.

Exemple

Considérons le système chariot dont le réseau de Petri du comportement attendu et le modèle de comportement attendu MOD_{CA} sont donnés figure 6.3.

Le marquage initial de ce réseau est $M_0 = [10000]^T$. Le système envoie au procédé une commande produisant l'événement b. Après vérification, cette observation est trouvée cohérente avec le modèle du comportement attendu. En faisant évoluer le modèle, le marquage atteint avec le franchissement de la transition sensibilisée associée à l'événement b est $M_k = [01000]^T$.

Supposons qu'une deuxième commande envoyée au procédé produise l'événement a. Après vérification, cette dernière observation est jugée incohérente par rapport à MOD_{CA}.

La recherche des différentes matrices de Rétablissement de Cohérence potentielles est initiée :

1. Le réseau de Petri à modifier est celui du comportement attendu (RdP_{CA}) ; la transition incohérente est $t_{inc} = t_4$; le marquage courant est $M_k = [01000]^T$; l'ensemble des places marquées de RdP_{CA} est $PM_k = \{p2\}$.
2. L'ensemble des nouvelles places d'entrée potentielles de la transition à l'origine de la détection d'incohérence t_4 est $M_{poss} = \{\emptyset, \{p2\}\} = \{Ep_1, Ep_2\}$. l'ensemble vide n'est pas une solution à notre problème. La seule nouvelle place d'entrée à t_4 est donc $p2$.

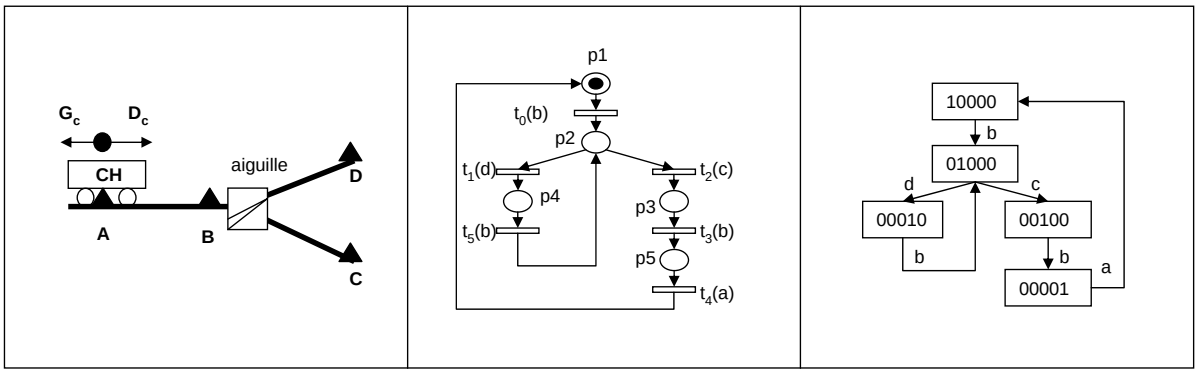


FIG. 6.3 – Système chariot, RdP_{CA} et MOD_{CA}

3. L'ensemble des vecteurs des places d'entrée à t_4 possibles est $Pre_{poss} = Pre_2(., t_4) = [01000]^T$.

4. L'ensemble C_{poss} des vecteurs d'incidence associés à la transition t_4 est $C_{poss} = \{C_2(., t_4) = Post(., t_4) - Pre_2(., t_4) = [10000]^T - [01000]^T = [1(-1)000]^T\}$.

5. Le réseau de Petri RdP_{CA} a un seul invariant de places dont nous ne détaillerons pas ici le calcul. Le support de cet invariant est $V_{P1} = [11111]^T$.

   Nous pouvons vérifier que $V_{P1}^T \bullet C_2(., t_4) = 0$ ce qui signifie que cette modification conserve l'invariant du RdP_{CA} initial. Donc l'ensemble des modifications possibles conservant les invariants est $E_{sol} = EI_1 = \{C_2(., t_4)\}$

6. Comme nous n'avons sur cet exemple qu'une seule modification possible, nous n'obtenons qu'une seule matrice de Rétablissement de Cohérence :

$$C_{RC} = [C(., t_0)C(., t_1)C(., t_2)C(., t_3)C_2(., t_4)C(., t_5)]$$

$$C_{RC} = \begin{pmatrix} -1 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & -1 & -1 & 0 & \mathbf{-1} & 1 \\ 0 & 0 & 1 & -1 & \mathbf{0} & 0 \\ 0 & 1 & 0 & 0 & \mathbf{0} & -1 \\ 0 & 0 & 0 & 1 & \mathbf{0} & 0 \end{pmatrix}$$

   Nous allons vérifier que la modification effectuée conserve les invariants de transitions du RdP_{CA} original sauf bien entendu les invariants dans lesquels la transition incohérente t_4 intervient.

   Le réseau initial RdP_{CA} avait deux invariants de transitions donnés respectivement par les séquences de tir $\sigma_1 = t_0t_2t_3t_4$ et $\sigma_2 = t_1t_5$ de supports respectifs $V_{T1} = [101110]^T$ et $V_{T2} = [010001]^T$

   Seul l'invariant σ_2, dans lequel t_4 n'intervient pas, est conservé dans le réseau modifié. ($C_{RC} \bullet V_{T1} = [101110]^T \neq 0$ et $C_{RC} \bullet V_{T2} = [010001]^T = 0$)

   La modification a par ailleurs entraîné la création d'un nouvel invariant de transition donné par la séquence de tir t_0t_4

7. Le nouveau RdP_{CA} défini par la matrice de Rétablissement de Cohérence C_{RC} et son graphe des marquages accessibles (MOD_{CA}) sont donnés sur la figure 6.4.

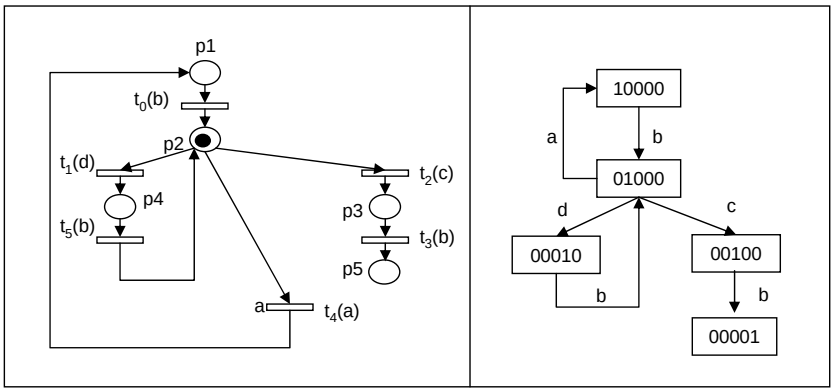


FIG. 6.4 – RdP_{CA} et MOD_{CA} après modification

Le comportement observé défini par la séquence d'événements observés ($S_{ev} = ba$) est maintenant présent dans MOD_{CA}.

Néanmoins, avec l'introduction de ce nouveau comportement dans MOD_{CA} nous avons créé une séquence de transitions $\sigma = t_0 t_2 t_3$ franchissable qui nous amène vers une situation de blocage. Il est dans ce cas possible que la composante pendante associée à σ traduise une erreur de modélisation.

6.5 Pertinence de la méthode par rapport aux erreurs du modèle

1. **Erreurs de modélisation.** Une erreur de modélisation dans un modèle peut correspondre, comme nous l'avons déjà présenté, à un surplus d'informations dans le modèle ou au contraire à un déficit d'informations.

   (a) **Excédent d'informations dans le modèle.** Un ou plusieurs comportements sont de trop dans le modèle original du système. Dans ce cas, ces comportements ne correspondent pas à des comportements normaux du système. La méthode de modification que nous proposons permet alors d'isoler la partie du modèle correspondant à cette erreur de modélisation.

   (b) **Déficit d'informations dans le modèle.** Il s'agit de comportements normaux qui ne sont pas décrits par le modèle original. Notre méthode de modification n'est pas la plus appropriée pour représenter le comportement observé car cette méthode modifie les comportements existants du modèle pour représenter le comportement observé au lieu d'ajouter directement le nouveau comportement. Le problème dans ce cas, est que la modification apportée au modèle risque de modifier certains comportements normaux du procédé. Il nous semble donc que pour une erreur de modélisation correspondant à un déficit d'informations il est plus approprié de modifier la structure du réseau en modifiant la dimension de C, de manière à ne rétablir la cohérence qu'en rajoutant le comportement observé.

2. **Dysfonctionnement du système.** Dans le cas où l'incohérence indique un dysfonctionnement du système, notre méthode de diagnostic par modification des modèles absorbera ce dysfonctionnement en l'intégrant au modèle. Après cette modification, une observation indiquant à nouveau ce comportement ne génèrera plus d'incohérence. Cependant s'il est possible d'identifier ce comportement comme un vrai symptôme de défaillance, son observation indiquera que le système rentre dans une partie du modèle qui modélise les défaillances affectant le système. Ce sont les résultats obtenus par l'identification de configurations qui vont nous permettre de conclure s'il s'agit d'un vrai ou faux symptôme.

Conclusions

Nous avons présenté dans le chapitre notre approche de diagnostic, qui consiste au rétablissement de la cohérence entre le comportement observé et les modèles après la détection d'une rupture de cohérence. Pour effectuer ce rétablissement de cohérence nous mettons en œuvre une méthode dont le but est de modifier le modèle du système pour représenter le comportement observé. Comme nous l'avons mentionné dans ce chapitre, en utilisant les réseaux de Petri, plusieurs méthodes sont envisageables pour réaliser la modification du modèle. La méthode que nous avons proposée modifie les coefficients de la matrice d'incidence C du RdP. Cette méthode conserve les propriétés du système exprimées via les invariants linéaires de places et de transitions.

Ce chapitre conclut la présentation de notre approche de détection et diagnostic de défaillances basée cohérence pour les systèmes à événements discrets.

Le prochain chapitre est un chapitre récapitulatif consacré à un exemple d'application de nos travaux de recherche.

Chapitre 7

Application de l'approche de diagnostic à base de cohérence

Introduction

Dans ce chapitre nous allons mettre en œuvre notre approche de diagnostic à base de cohérence. Cette approche va être appliquée à une maquette de tri de pièces conçue par l'entreprise FESTO. Cette maquette est utilisée à l'Institut National de Sciences Appliqués (INSA) à Toulouse.

Nous allons commencer le chapitre par la description du système sur lequel nous allons appliquer notre approche. Suite à cette description, nous allons modéliser ce système, en présentant les modèles de comportement attendu MOD_{CA}, le modèle de la commande MOD_C. Il est réaliste dans ce cas, de supposer disposer, du modèle de comportement réel de ce système MOD_{CR}. Les observations issues du fonctionnement de ce système seront modélisées par le modèle d'observation MOD_{OBS}. La modification des modèles, l'étape de diagnostic suite à une détection d'une incohérence, est illustrée. Nous montrons à la fin de ce chapitre comment les observations sont utilisées pour parcourir le graphe d'identification en ligne et ainsi identifier les configurations des trois modèles du système.

7.1 Description du système

La maquette a pour but d'assurer le déplacement de pièces stockées dans un magasin vers une station de tri qui réalise la sélection des pièces en fonction de leur type. Trois types de pièces peuvent être triées par ce système : des pièces métalliques, des pièces rouges et des pièces noires. Cette maquette est un modèle d'étude d'un équipement industriel de la société FESTO [FESTO,]. Elle est présentée dans la figure 7.1.

7.1.1 Composants du système

La maquette est composée de quatre modules différents : le module magasin, le module préhenseur, le module convoyeur et le module tri.

1. **Le module magasin.** Ce module est composé de deux éléments : un stock de pièces et un éjecteur. Le stock de pièces est un tube cylindrique dans lequel les

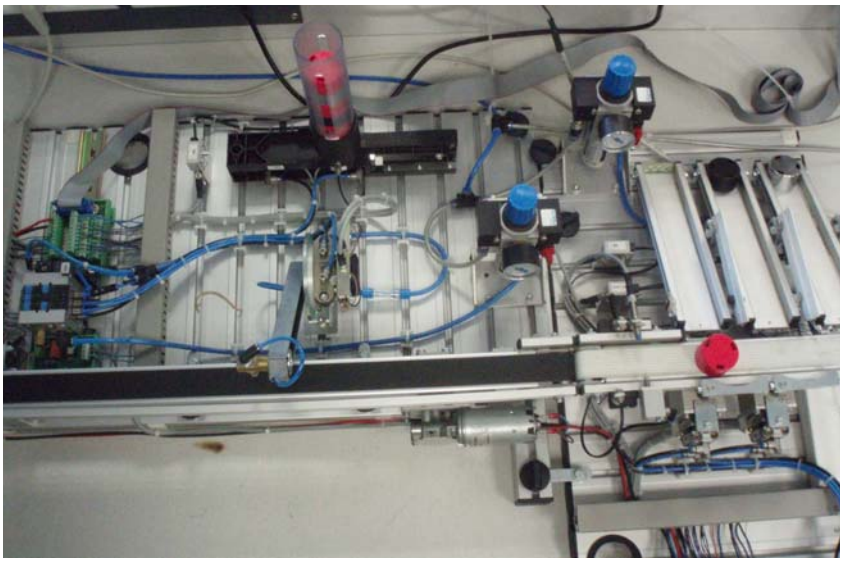


FIG. 7.1 – Présentation de la maquette de tri FESTO

pièces sont empilées manuellement. Un capteur de position identifié par p_{mag} permet détecter la présence d'une pièce dans le magasin. L'éjecteur de pièces est constitué par un vérin pneumatique, qui permet d'éjecter une pièce dehors du magasin. Les pièces éjectées hors du magasin sont détectées par le capteur pd_{mag}. La commande associée à l'activation du vérin est la commande V_{ON}. La position rentrée du vérin est signalée par le capteur deb_v et la position sortie du vérin est indiquée par le capteur fin_v. Ce module est représenté dans la figure 7.2.

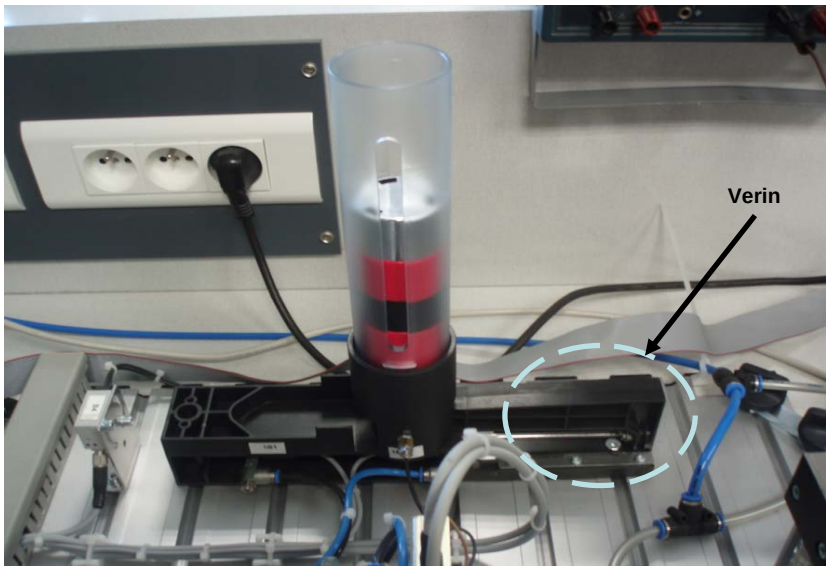


FIG. 7.2 – Le module magasin

2. **Le module préhenseur.** Ce module est constitué d'un préhenseur de type ventouse placé à l'extrémité d'un bras. La fonction de ce préhenseur est de déplacer une pièce du module magasin vers le module convoyeur. Deux commandes permettent de réaliser ce déplacement :

– la commande $CTOM$ permet de déplacer le préhenseur du convoyeur vers le module magasin,
– la commande $MTOC$ permet de déplacer le préhenseur du magasin vers le convoyeur.

La position du bras coté du magasin est signalée par le capteur pr_{mag} et la position du bras coté du convoyeur est détectée par pr_{con}. Deux commandes sont utilisées pour saisir la pièce :

– la commande $VIDE$ permet de faire le vide sous la ventouse et de capturer ainsi la pièce,
– la commande $RELACHE$ remet la ventouse à la pression atmosphérique relâchant ainsi la pièce.

Le capteur pr_{asp} permet de détecter qu'une pièce a été capturée par la ventouse. La figure 7.3 nous montre ce module.

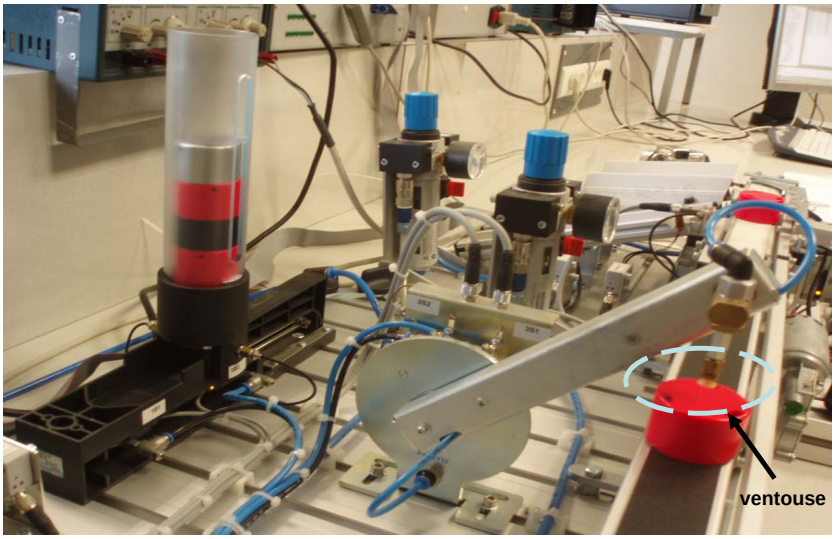


FIG. 7.3 – Le module Préhenseur

3. **Le module convoyeur.** Ce module est composé d'un tapis roulant dont la fonction est d'assurer le transport des pièces vers le module de tri. Le capteur p_{con} identifie une pièce dans le convoyeur. La commande utilisée dans ce module pour mettre en marche le convoyeur est la commande C_{ON}. Le convoyeur est montré la figure 7.3.
4. **Le module de tri.** Ce module chargé de réaliser le tri des différents types de pièces est constitué de quatre éléments :
   – un tapis roulant,
   – la goulotte 1,
   – la goulotte 2,
   – la goulotte 3.

   Les pièces sur le tapis sont détectées par le capteur p_{tap}. La fonction du tapis roulant est de transporter les pièces vers la zone de tri puis vers les goulottes. Les pièces sont stockées dans les différentes goulottes le tri. La commande T_{ON} se charge de mettre en mouvement le tapis de ce module. Lorsqu'une pièce arrive à l'entrée du tapis, elle est systématiquement bloquée par une butée pour permettre

l'identification de son type. La libération de la pièce est réalisée par la commande B_{OFF}, qui assure l'escamotage de la butée pendant deux secondes. Quand une pièce est bloquée par la butée, deux capteurs tout ou rien permettent d'identifier son type : le capteur p_{metal} signale la présence d'une pièce métallique et le capteur p_{autre} indique la présence d'une pièce noire ou d'une pièce rouge. Pour aiguiller les pièces sur les goulottes correspondantes, le tapis est équipé de deux barrages, le barrage $B1$ et le barrage $B2$. La commande $B1_{ON}$ permet d'aiguiller les pièces vers la goulotte 1 utilisée pour stocker les pièces métalliques. La commande $B2_{ON}$ permet d'aiguiller les pièces vers la goulotte 2 destinée à stocker les pièces rouges. Si aucun barrage n'est activé la pièce est orientée vers la goulotte numéro 3 destinée à stocker les pièces noires.

Chaque goulotte est équipée à son entrée d'un capteur qui permet de détecter le passage d'une pièce. Le capteur de la goulotte 1 est noté p_{g1},celui de la goulotte 2 p_{g2}, et celui de la troisième goulotte p_{g3}. Les goulottes ont une capacité limitée à 5 pièces. Ce module est illustré figure 7.4.

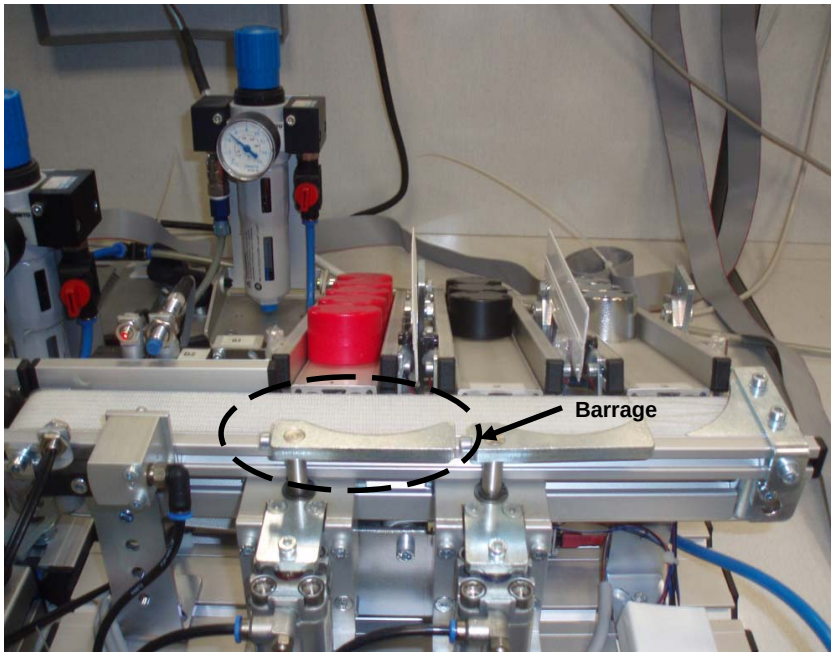


FIG. 7.4 – Le module tri

7.1.2 Cahier des charges

Module magasin

A l'état initial, le vérin est en position rentrée et il ne doit évacuer une pièce que si la pièce précédente a déjà été dégagée par le préhenseur.

Module préhenseur

A l'état initial, le préhenseur est du côté convoyeur. Son fonctionnement est requis lorsqu'une pièce a été éjectée par le vérin du magasin. La commande $VIDE$ doit être maintenue tant que la pièce n'est pas déposée sur le convoyeur. Cette opération de dépose d'une pièce sur le convoyeur ne peut avoir lieu que si ce dernier est à l'arrêt.

Observation (o_i)	événement ($Eo(o_i)$)	interprétation
Module magasin		
p_{mag}	a_1	présence d'une pièce dans le magasin
pd_{mag}	a_2	présence d'une pièce en dehors du magasin
deb_v	a_3	vérin en position rentrée
fin_v	a_4	vérin en position sortie
Module préhenseur		
pr_{con}	b_1	présence du préhenseur coté convoyeur
pr_{mag}	b_2	présence du préhenseur coté magasin
pr_{aps}	b_3	pièce saisie par la ventouse du préhenseur
Module convoyeur		
p_{con}	c_1	présence d'une pièce dans le convoyeur
Module tri		
p_{tap}	d_1	pièce en zone d'identification
p_{autre}	d_2	pièce rouge ou noire
p_{metal}	d_3	pièce métallique
p_{g1}	d_4	pièce en entrée de goulotte 1
p_{g2}	d_5	pièce en entrée de goulotte 2
p_{g3}	d_6	pièce en entrée de goulotte 3

TAB. 7.1 – Définition de la fonction Eo

Module convoyeur

A l'état initial, le convoyeur est à l'arrêt et il est mis en marche lors de la première dépose de pièce. Quand une pièce doit être déposée, le convoyeur doit s'arrêter. La dépose terminée, le convoyeur est remis en marche.

Module de tri

A l'état initial, le tapis est à l'arrêt. Il est mis en marche lorsqu'une pièce est déposée sur son entrée. Une pièce peut être dirigée vers sa goulotte à condition que cette dernière ne soit pas saturée.

7.2 Modélisation du système

Deux modèles sont considérés : le modèle de la commande et le modèle du comportement attendu. Nous allons commencer par définir quels sont les événements associés à chaque module du système. Ces événements attendus sont directement liés aux signaux envoyés par les capteurs associés à chaque module. Le tableau 7.1 montre la correspondance des signaux issus des capteurs (les observations) avec les événements observables de ce système. Il s'agit en fait de la définition de la fonction Eo introduite au chapitre 4.

7.2.1 Le modèle de comportement attendu MOD_{CA}

Le comportement attendu est basé sur les événements observables en fonctionnement normal lors de l'application d'une commande. Ainsi, l'ensemble des événements

observables du système correspondant aux signaux des capteurs est le suivant :

$$E_{obs} = \{a_1, a_2, a_3, a_4, b_1, b_2, b_3, c_1, d_1, d_2, d_3, d_4, d_5, d_6\}$$

La figure 7.5 montre ce comportement attendu.

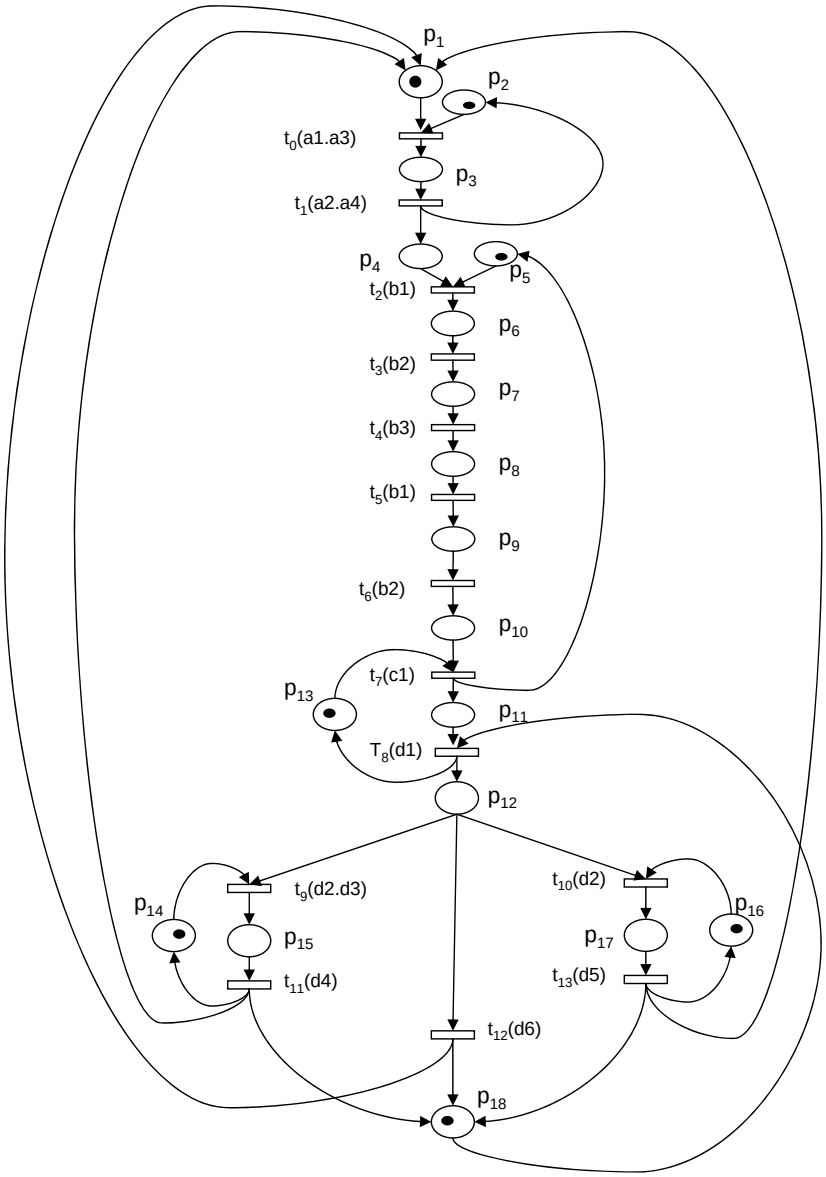


FIG. 7.5 – RdP du comportement attendu

Les transitions de RdP_{CA} sont étiquetées par des événements observables du système comme indiqué dans le tableau 7.2 :

Les matrices d'incidence et le graphe des marquages accessibles qui comporte 12 états notés M_0 à M_{11} sont fournis en annexe page 159.

7.2.2 Les propriétés du RdP_{CA}

L'analyse à l'aide de l'outil TINA [LAAS-CNRS,] montre que ce réseau possède les trois bonnes propriétés : vivant, borné et réinitialisable.

transition (t_i)	événement associé ($Ev(t_i)$)
t_0	$(a_1.a_3)$
t_1	$(a_2.a_4)$
t_2	(b_1)
t_3	(b_2)
t_4	(b_3)
t_5	(b_1)
t_6	(b_2)
t_7	(c_1)
t_8	(d_1)
t_9	$(d_2.d_3)$
t_{10}	(d_2)
t_{11}	(d_4)
t_{12}	(d_6)
t_{13}	(d_5)

TAB. 7.2 – définition de la fonction Ev pour MOD_{CA}

L'analyse structurelle met en évidences les composantes conservatives et répétitives stationnaires.

Les composantes conservatives sont au nombre de sept

* IP_1 :$M(p_1) + M(p_3) + M(p_4) + M(p_6) + M(p_7) + M(p_8) + M(p_9) + M(p_{10}) + M(p_{11}) + M(p_{12}) + M(p_{15}) + M(p_{17}) = 1$
* IP_2 :$M(p_5) + M(p_6) + M(p_7) + M(p_8) + M(p_9) + M(p_{10}) = 1$
* IP_3 :$M(p_{12}) + M(p_{15}) + M(p_{17}) + M(p_{18}) = 1$
* IP_4 :$M(p_2) + M(p_3) = 1$
* IP_5 :$M(p_{16}) + M(p_{17}) = 1$
* IP_6 :$M(p_{14}) + M(p_{15}) = 1$
* IP_7 :$M(p_{11}) + M(p_{13}) = 1$

Les invariants linéaires de transitions sont au nombre de trois

* IT_1 :$t_0\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_8\ t_9\ t_{11}$
* IT_2 :$t_0\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_8\ t_{10}\ t_{13}$
* IT_3 :$t_0\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_8\ t_{12}$

Maintenant, nous allons donner la modélisation de la commande pour cette maquette de tri de pièces.

7.2.3 Le modèle de la commande MOD_C

La figure 7.6 montre le réseau de la commande. Le graphe des marquages associé à ce réseau est appelé modèle de commande MOD_C.

Les transitions de RdP_C sont étiquetées par des événements observables du système comme décrit dans le tableau 7.3 (ϵ signale une transition interne du réseau, c'est à dire modélisant une évolution non liée à l'environnement comme la réservation d'une ressource).

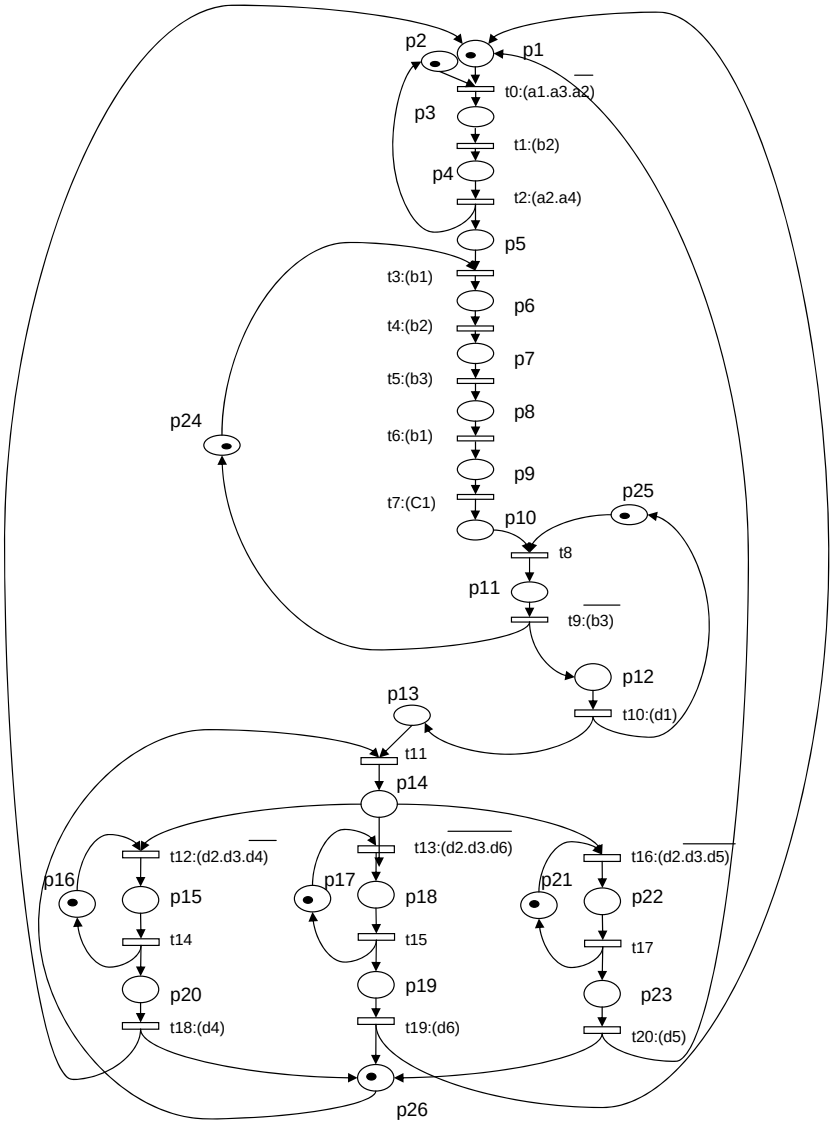


FIG. 7.6 – RdP de la commande

Les matrices d'incidence et le graphe des marquages accessibles qui comporte 19 états notés M_0 à M_{18} sont fournies en annexe page 162.

7.2.4 Les propriétés du RdP_C

Comme précédemment, l'analyse à l'aide de l'outil TINA [LAAS-CNRS,] montre que ce réseau possède les trois bonnes propriétés : vivant, borné et réinitialisable.

L'analyse structurelle met en évidence les composantes conservatives et répétitives stationnaires.

Les composantes conservatives sont au nombre de huit

* IP_1 : $M(p_1) + M(p_3) + M(p_4) + M(p_5) + M(p_6) + M(p_7) + M(p_8) + M(p_9) + M(p_{10}) + M(p_{11}) + M(p_{12}) + M(p_{13}) + M(p_{14}) + M(p_{15}) + M(p_{18}) + M(p_{19}) + M(p_{20}) + M(p_{22}) + M(p_{23}) = 1$
* IP_2 : $M(p_{14})+M(p_{15})+M(p_{18})+M(p_{19})+M(p_{20})+M(p_{22})+M(p_{23})+M(p_{26}) = 1$

transition (t_i)	événement associé ($Ev(t_i)$)
$t0$	$(a1.a3.\overline{a2})$
$t1$	$(a2.a4)$
$t2$	$(b1)$
$t3$	$(b2)$
$t4$	$(b3)$
$t5$	$(b1)$
$t6$	ϵ
$t7$	$(b3)$
$t8$	$(d1)$
$t9$	ϵ
$t10$	$(d2.d3.\overline{d4})$
$t11$	ϵ
$t12$	$(d4)$
$t13$	$(\overline{d2.d3.d6})$
$t14$	ϵ
$t15$	$(d6)$
$t16$	$(d2.\overline{d3.d5})$
$t17$	ϵ
$t18$	$(d5)$

TAB. 7.3 – définition de la fonction Ev pour MOD_C

* $IP_3 : M(p_6) + M(p_7) + M(p_8) + M(p_9) + M(p_{10}) + M(p_{11}) + M(p_{24}) = 1$
* $IP_4 : M(p_2) + M(p_3) + M(p_4) = 1$
* $IP_5 : M(p_{11}) + M(p_{12}) + M(p_{25}) = 1$
* $IP_6 : M(p_{21}) + M(p_{22}) = 1$
* $IP_7 : M(p_{17}) + M(p_{18}) = 1$
* $IP_8 : M(p_{15}) + M(p_{16}) = 1$

Les composantes répétitives stationnaires sont au nombre de trois

* $IT_1 : t_0\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_8\ t_9\ t_{10}\ t_{11}\ t_{12}\ t_{14}\ t_{18}$
* $IT_2 : t_0\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_8\ t_9\ t_{10}\ t_{11}\ t_{13}\ t_{15}\ t_{19}$
* $IT_3 : t_0\ t_1\ t_2\ t_3\ t_4\ t_5\ t_6\ t_7\ t_8\ t_9\ t_{10}\ t_{11}\ t_{16}\ t_{17}\ t_{20}$

7.3 Application de l'approche

Dans cette section nous allons appliquer notre approche de détection et de diagnostic étape par étape à cet exemple. Pour cela, nous allons étudier une séquence d'observations et les événements associés en nous focalisant sur les observations les plus intéressantes pour notre approche. Ces observations vont nous permettre de montrer toutes les étapes de notre approche : la détection, le diagnostic et l'identification de la configuration des modèles (MOD_{CA} et MOD_C).

Tous d'abord, nous allons commencer par mettre en place le modèle d'observation utilisé dans l'étape de détection.

7.3.1 Analyse des observations

7.3.1.1 Modélisation du comportement observé avec MOD_{OBS}

A l'initialisation, les premières observations concernent les capteurs p_{mag} et deb_v associés respectivement aux événements a_1 et a_3 dont la conjonction constitue la condition de la transition t_0 de MOD_{CA}. Nous allons montrer la construction du MOD_{OBS} selon les possibilités générales décrites dans le chapitre 4. Deux trajectoires sont construites à la modélisation de cette première observation :

1. $T_{Ra} = M_0^{OBS} \xrightarrow{t1(a1.a3)} M_0^{OBS}$ (possibilité P_1)
2. $T_{Ra} = M_0^{OBS} \xrightarrow{t1(a1.a3)} M_x^{OBS}$ (possibilité P_3)

où la valeur M_x est inconnue à cet état de la modélisation. La construction d'arcs de type P_2 n'est pas applicable pour cette première observation, l'ensemble des états étant réduit au singleton état initial. L'ensemble de marquages ou états contenus dans ce modèle est $M_{OBS} = \{M_0, M_x\}$ et l'ensemble de transitions que contient MOD_{OBS} est $T_{OBS} = \{t1(a1.a3)\}$. La représentation graphique de ce modèle est donnée par la figure 7.7.

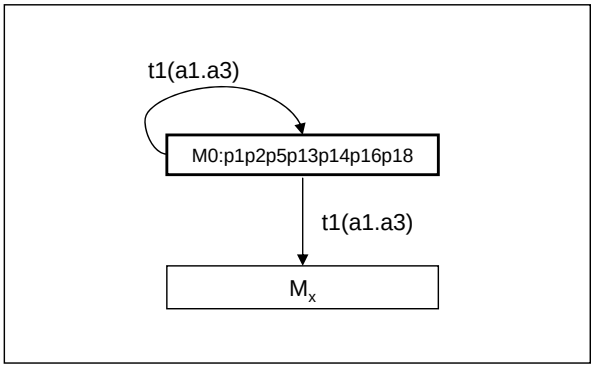


FIG. 7.7 – MOD_{OBS} après les premières observations

Nous allons effectuer l'étape de détection par la vérification de cohérence entre l'événement observé et les modèles du système. Nous allons commencer cette vérification de cohérence par le modèle de comportement attendu MOD_{CA}.

7.3.1.2 La détection par vérification de cohérence

Vérification de cohérence entre MOD_{OBS} et MOD_{CA}

Le produit cartésien entre les deux modèles est utilisé pour réaliser la vérification. Le graphe de vérification $GVER^{CA}$ généré par le produit cartésien de MOD_{OBS} et MOD_{CA} contient 24 états et 25 événements provoquant les transitions entre ces états.

Nous allons évaluer chaque possibilité représentée dans le modèle d'observations. La figure 7.8 donne une représentation partielle de ce produit cartésien. La représentation partielle montrée $GVER^{CA}$ permet de réaliser la vérification de cohérence. En analysant chaque possibilité réalisée avec le modèle d'observation, nous cherchons la synchronisation des comportements décrits dans les deux modèles pour déterminer la cohérence ou l'incohérence de cette première observation. Nous cherchons à prouver si l'événement

observé fait évoluer le modèle à partir du même état de départ vers des états d'arrivée équivalents pour les deux modèles, selon chaque possibilité :

P_1 :

$$M_0^{OBS}, M_0^1 \xrightarrow{(a1.a3)^{OBS},(a1.a3)^1} M_0^{OBS}, M_1^1$$

Dans cette première possibilité nous cherchons à vérifier si MOD_{CA} reste dans le même état de départ avec l'événement observé. Comme ce n'est pas le cas, la seconde possibilité d'évolution représentée dans MOD_{OBS} est alors vérifiée.

P_3 :

$$M_0^{OBS}, M_0^1 \xrightarrow{(a1.a3)^{OBS},(a1.a3)^1} M_x^{OBS}, M_1^1$$

Dans le modèle des observations, M_x représente un état qui n'a pas déjà été parcouru pendant l'observation. La transition ci-dessus est donc une candidate pour expliquer l'observation et les événements qui lui sont associés. Nous en concluons que M_x est l'état M_1 de MOD_{CA}. Ceci montre qu'il existe une transition et un état dans MOD_{CA} cohérent avec la possibilité P_3 d'évolution inscrite précédemment dans MOD_{OBS}. La possibilité P_3 est alors retenue dans MOD_{OBS} et toutes les autres possibilités sont éliminées (ici uniquement P_1). Reste à vérifier la cohérence entre MOD_{OBS} et MOD_C.

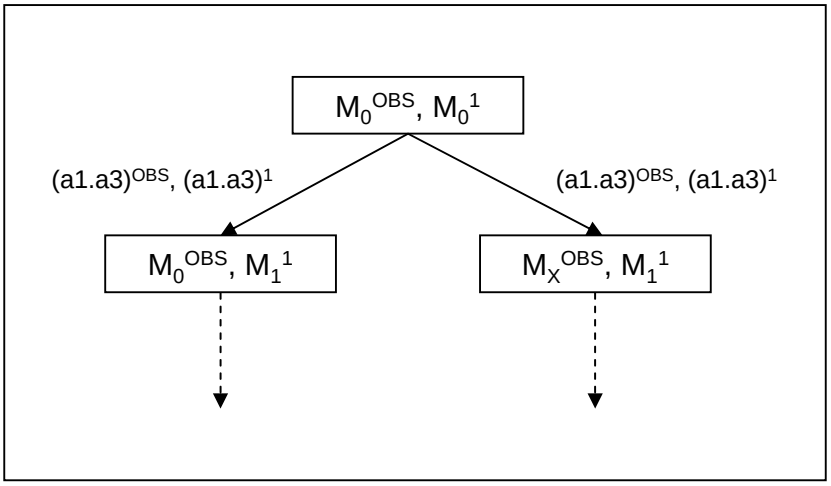


FIG. 7.8 – Vue partielle du graphe de vérification de MOD_{CA}

Vérification de cohérence entre MOD_{OBS} et MOD_C

A partir de l'état initial du modèle de la commande, nous allons vérifier si l'observation correspond au compte rendu attendu par la commande. Nous allons utiliser à nouveau le produit cartésien entre MOD_{OBS} et le MOD_C pour vérifier la cohérence, mais en restreignant cette fois MOD_{OBS} au comportement cohérent avec MOD_{CA} déterminé précédemment (seule la transition P_3 a été conservée).

Une vue partielle du graphe obtenu en réalisant le produit cartésien des deux modèles est donnée figure 7.9. Dans ce graphe, la transition

$$M_0^{OBS}, M_0^2 \xrightarrow{(a1.a3)^{OBS},(a1.a3.\overline{a2})^2} M_1^{OBS}, M_1^2$$

est cohérente avec la possibilité P_3 de l'observation.

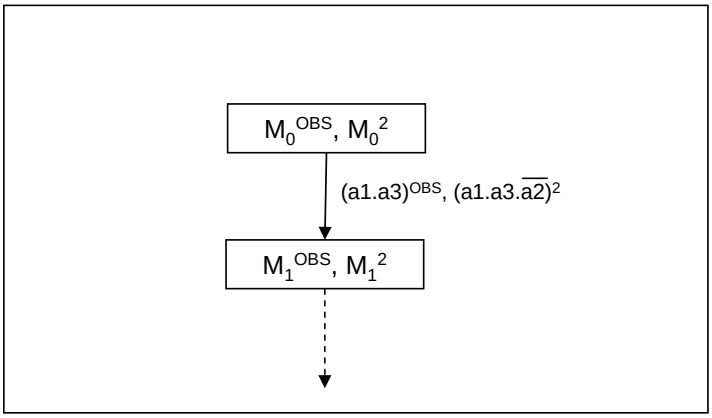


FIG. 7.9 – Vue partielle du graphe de vérification de MOD_C

7.3.1.3 L'identification de la configuration

Cette observation nous donne des informations utiles pour l'identification de la configuration entre les modèles du système MOD_{CA} et MOD_{CR}. Nous avons constaté lors de l'observation, après l'étape de vérification de cohérence, que cette observation est cohérente avec MOD_{CA} et aussi avec MOD_C. La valeur de la constatation associée à la trajectoire cohérente T_{Rb} vaut donc

$$CNS_{ARC}(T_{Rb}) = (1, ?, 1)$$

Pour déterminer si cette constatation doit être considérée comme un symptôme de défaillance, nous devons faire une hypothèse concernant sa cohérence avec le modèle du comportement réel du système noté MOD_{CR}. Deux cas sont possibles :

1. L'observation est cohérente avec MOD_{CR}. Cette supposition nous permet d'obtenir la valeur de la constatation entièrement définie $CNS_{ARC}(T_{Rb}) = (1, 1, 1)$. Cette constatation permet d'affirmer que les trois modèles ont des comportements communs, dont fait partie la trajectoire T_{Rb}.
2. L'observation est incohérente avec MOD_{CR}. Cette supposition nous permet d'obtenir la valeur de la constatation $CNS_{ARC}(T_{Rb}) = (1, 0, 1)$ nous indiquant explicitement que les deux modèles MOD_{CA} et MOD_C ont des comportements communs, mais qu'il en existe au moins un, le dernier observé, qui ne correspond pas au fonctionnement réel du procédé. En d'autres termes, cela indique que la trajectoire T_{Rb} est décrite dans les deux modèles MOD_{CA} et MOD_C par erreur.

Remarque : les deux suppositions réalisées par rapport à MOD_{CR} n'altèrent en rien la connaissance apportée par l'observation par rapport à MOD_{CA} et MOD_C. Elles permettent de resituer nos travaux dans le contexte habituel des approches de détection ou de diagnostic à base de modèles qui font **toutes** l'hypothèse que MOD_{CR} est exactement connu...

Pour simplification nous allons présenter à la fin de ce chapitre le parcours du graphe d'identification de configuration, pour toutes les observations utilisées dans cette exemple d'application.

La suite du traitement concerne le diagnostic consistant en la modification des modèles.

7.3.1.4 Le diagnostic

Dans la situation que nous venons de décrire, la modification des modèles pour le rétablissement de la cohérence (le diagnostic) n'est pas nécessaire car l'observation est cohérente avec MOD_{CA} et MOD_C. Cependant nous pouvons tirer des conclusions basées sur les différentes valeurs des constatations pour l'observation :

- la valeur de la constatation $CNS_{ARC}(T_{Rb}) = (1, 1, 1)$ nous permet d'indiquer que l'événement observé correspond à un fonctionnement normal du système car cet événement appartient à MOD_{CR}.
- la valeur de la constatation $CNS_{ARC}(T_{Rb}) = (1, 0, 1)$ nous permet également de déduire que le comportement observé est en réalité un dysfonctionnement du système car ce comportement sort du comportement décrit dans MOD_{CR}. Étant donné que MOD_C et MOD_{CA} ont été vérifiés cohérents avec l'observation, nous pouvons conclure que ces deux modèles contiennent des erreurs (description du comportement observé) qui masquent un dysfonctionnement du système non détectable à partir de la connaissance de ces deux modèles uniquement.

Nous allons présenter dans la suite de ce document, d'autres scénarii qui nous permettrons d'illustrer entièrement les différentes étapes de nos propositions, de la détection jusqu'à la modification des modèles.

Pour simplification nous nous limitons aux événements de l'exemple d'application qui nous amènent à l'utilisation des différentes méthodes. Ces observations présentées dans ce chapitre nous apportent aussi des constatations qui vont nous permettre de progresser également dans l'identification de la configuration des ensembles.

7.3.2 Analyse de la seconde observation

Après la réception de la première observation le système de commande envoie au procédé la commande V_{ON} pour sortir la tige du vérin. L'observation issue du procédé après l'application de la commande concerne les capteurs fin_v et p_{mag}. Ces observations correspondent aux événements a_2 et a_4 associés à la transition t_2 de MOD_{CA}.

Décrivons en détails les étapes suivantes du traitement de ces observations.

7.3.2.1 Modélisation du comportement observé MOD_{OBS}

Le modèle d'observation s'incrémente à la réception de cette observation. Étant donné la vérification de cohérence avec la première observation, MOD_{OBS} contient l'ensemble d'états donné par $M_{OBS} = \{M_0, M_1\}$. Maintenant, l'état courant M_c est représenté par l'état M_1 et c'est à partir de cet état que nous allons modéliser la deuxième observation. La figure 7.10 représente l'intégration de cette nouvelle observation à MOD_{OBS}. Trois transitions sont créées à la modélisation de cette observation.

1. P_1 : $T_{Rc1} = M_1^{OBS} \xrightarrow{t2(a2.a4)} M_1^{OBS}$
2. P_2 : $T_{Rc2} = M_1^{OBS} \xrightarrow{t2(a2.a4)} M_0^{OBS}$
3. P_3 : $T_{Rc3} = M_1^{OBS} \xrightarrow{t2(a2.a4)} M_x^{OBS}$

L'ensemble de marquages ou états contenus dans ce modèle est maintenant $M_{OBS} = \{M_0, M_1, M_x\}$ où M_x a une valeur inconnue à cet état de modélisation. L'ensemble des transitions que contient MOD_{OBS} est T_{OBS}={t1, t2}.

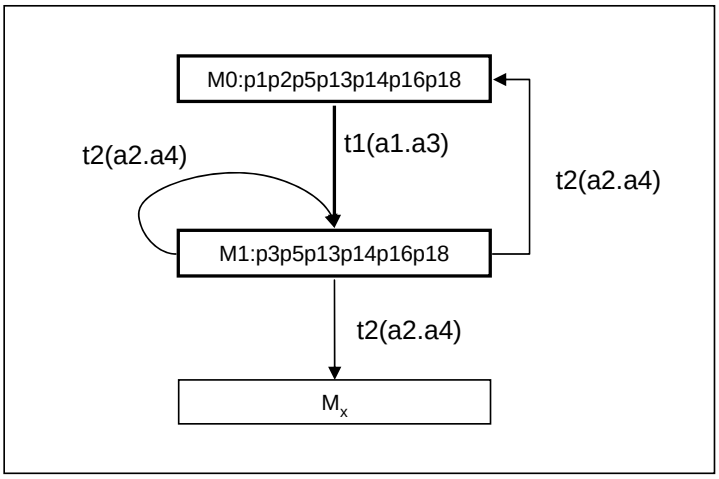


FIG. 7.10 – MOD_{OBS} après modélisation de la deuxième observation

7.3.2.2 La détection par vérification de cohérence

La vérification de la cohérence avec MOD_{CA}

Le graphe de vérification $GVER^{CA}$ issu du produit cartésien de MOD_{OBS} et MOD_{CA} est composé de 36 états et de 38 événements. Une représentation partielle de ce graphe est montrée dans la figure 7.11. Nous vérifions la cohérence pour chaque possibilité de MOD_{OBS} :

1. P_1 : $M_1^{OBS}, M_1^1 \xrightarrow{(a2.a4)^{OBS},(a2.a4)^1} M_1^{OBS}, M_2^1$
   Étant donné que M_1^{OBS} est différent de M_2^1, les comportements ne sont pas cohérents, même si les deux modèles sont sensibles à cette observation, les évolutions des modèles sont différentes, car MOD_{OBS} ne change pas d'état tandis que MOD_{CA} change d'état.
   Nous vérifions maintenant la deuxième possibilité de MOD_{OBS}.
2. P_2 : $M_1^{OBS}, M_1^1 \xrightarrow{(a2.a4)^{OBS},(a2.a4)^1} M_0^{OBS}, M_2^1$
   Dans cette possibilité nous trouvons que M_2^1, n'est pas un état possédant les caractéristiques requises pour les évolutions de type P_3. Il n'est pas possible de vérifier cette possibilité, les comportements sont donc incohérents.
   Étant donné cette incohérence, nous passons à la vérification de la troisième possibilité.
3. P_3 :$M_1^{OBS}, M_1^1 \xrightarrow{(a2.a4)^{OBS},(a2.a4)^1,} M_x^{OBS}, M_2^1$
   Comme il est possible de l'observer dans le graphe des marquages de MOD_{CA}, l'événement $(a_2.a_4)$ fait évoluer le graphe de l'état M_1^1 vers l'état M_2^1 où $M_1^1 \neq M_2^1$ et étant donné que $M_2^1 \in M_{NF}$, cette possibilité 3 est bien vérifiée et ces comportements sont trouvés cohérents. A la vérification de cette possibilité, nous éliminons toutes les autres possibilités du MOD_{OBS}. La valeur M_x^{OBS} peut être déterminée, $M_x^{OBS} = M_2^1$.

Maintenant nous allons passer à la vérification de la cohérence de MOD_{OBS} et MOD_C.

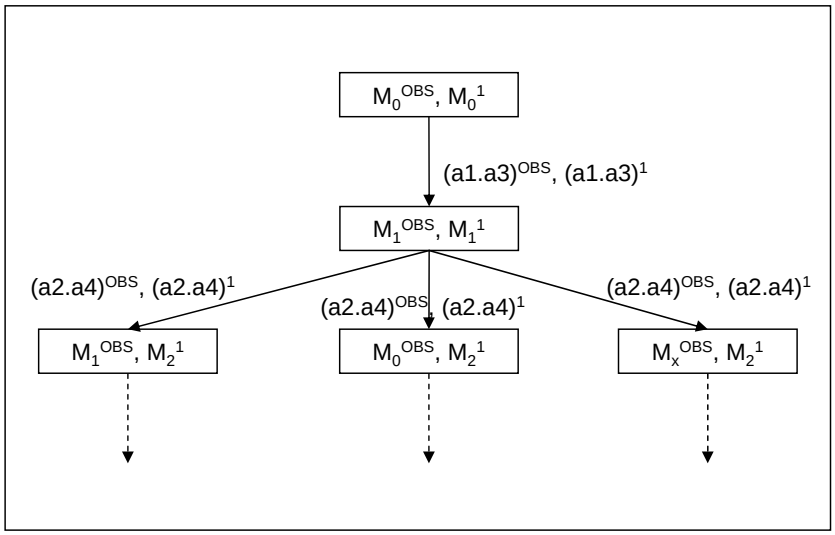


FIG. 7.11 – $GVER^{CA}$ pour la deuxième observation

Vérification de cohérence entre MOD_{OBS} et MOD_C

Nous effectuons ceci avec un modèle d'observation comportant uniquement la possibilité P_3 vérifiée précédemment par MOD_{OBS} et MOD_{CA}. La figure 7.12 montre une représentation partielle du graphe de vérification de cohérence $GVER^C$.

P_3 **:** ce que nous cherchons à savoir avec cette possibilité, c'est si l'état de la commande M_1^2 est sensible à l'événement observé et si, en plus, l'événement $(a_2.a_4)$ fait évoluer le modèle vers un état non visité auparavant pour les observations précédentes de la séquence observée. Étant donné que le seul état de MOD_C pouvant être atteint à partir de l'état courant M_1^2 est M_2^2 et que de plus M_1^2 n'est pas sensible à l'événement $(a_2.a_4)$

$$M_1^{OBS}, M_1^2 \xrightarrow{(a2.a4)^{OBS},(b1)^2} M_2^{OBS}, M_2^2$$

la deuxième observation correspondant à l'événement (a_2, a_4) est donc incohérente par rapport à MOD_C.

7.3.2.3 Identification de la configuration

La constatation associée à la trajectoire T_{Rc3} modélisant la deuxième observation fournit une connaissance différente de celle apportée par la première.

La valeur de cette constatation est $CNS_{ARC}(T_{Rc3}) = (1, ?, 0)$. Comme dans le scénario précédent, nous allons réaliser les suppositions respectives par rapport à MOD_{CR}.

1. $CNS_{ARC}(T_{Rc3}) = (1, 1, 0)$
   Cette première valeur de constatation qui résulte de l'hypothèse selon laquelle la deuxième observation (T_{Rc3}) est cohérente avec MOD_{CR}, nous indique l'existence d'un sous-ensemble type AR dans la configuration des modèles. Notons que cette valeur de constatation indique que le procédé n'a pas subi de défaillance, mais qu'une erreur de modélisation est présente dans MOD_C.

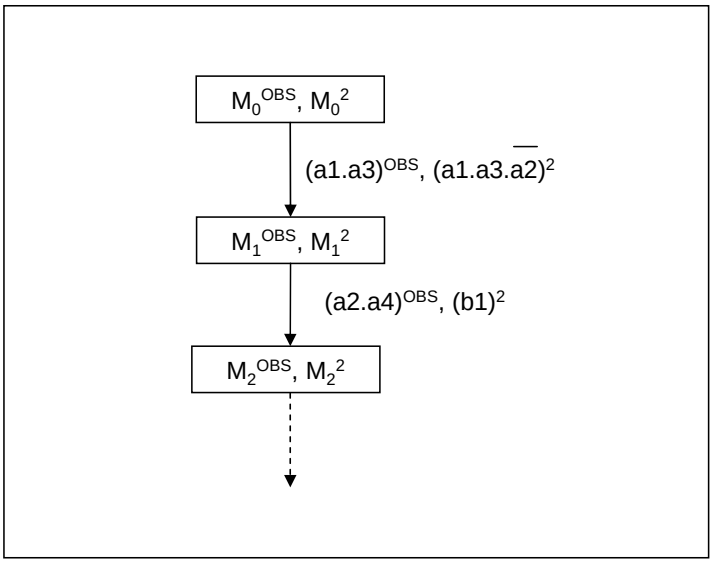


FIG. 7.12 – $GVER^C$ pour la deuxième observation

2. $CNS_{ARC}(T_{Rc3}) = (1, 0, 0)$
   La deuxième constatation suppose que la deuxième observation est incohérente avec MOD_{CR} et nous indique de l'existence d'un sous-ensemble type A dans notre configuration de modèles.

Cette connaissance vient s'ajouter à celle apportée par la première observation. En quelques mots et après ces deux observations, cela nous permet d'affirmer que les modèles ont une intersection, dans la première hypothèse MOD_C est inclus dans MOD_{CR} alors que dans la deuxième hypothèse MOD_C décrit des comportements qui ne sont pas dans le modèle MOD_{CR}. Nous reviendrons sur ce cumul de connaissance en fin du chapitre.

Ayant détecté une rupture de cohérence entre la deuxième observation et MOD_C, nous devons réaliser l'étape de diagnostic pour rétablir la cohérence des modèles.

7.3.2.4 Le diagnostic

La détection ayant indiqué que MOD_C devait être modifié, nous appliquons la démarche de diagnostic expliquée dans le chapitre 6, consistant à modifier des coefficients de la matrice d'incidence C.

1. Détermination de l'ensemble de places marquées de MOD_C dans l'état courant M_1^1 :

$$PM_k = \{p3, p16, p17, p21, p24, p25, p26\}$$

2. Calcul de l'ensemble des possibles nouvelles places d'entrée de la transition incohérente t_{inc}, où $t_{inc} = t2(a2.a4)$. Au total nous avons 128 (2^7) possibilités .
   M_{poss} = {$\emptyset$, {p_3}, {p_{16}}, {p_{17}}, {p_{21}}, {p_{24}}, {p_{25}}, {p_{26}}, {p_3, p_{16}}, {p_3, p_{17}}, {p_3, p_{21}}, {p_3, p_{24}},{p_3, p_{25}}, {p_3, p_{26}}, ... ,{p_3, p_{16}, p_{17}, p_{21}, p_{24}, p_{25}, p_{26} }
3. Construire l'ensemble de tous les vecteurs de places d'entrée possibles Pre_{poss} à la transition incohérente $t_{inc} = t2(a2.a4)$ en utilisant les sous-ensembles de places

de M_{poss}. Étant donné la taille du vecteur nous allons le présenter dans sa forme transposée :

$$\begin{aligned} Pre_1 &= [00100000000000000000000000]^T \\ Pre_2 &= [00000000000000010000000000]^T \\ Pre_3 &= [00000000000000001000000000]^T \\ Pre_{11} &= [00100000000000001000100000]^T \\ Pre_{128} &= [00100000000000011000100111]^T \end{aligned}$$

4. Composer les nouvelles colonnes de la matrice d'incidence C_{poss} pour la transition incohérente t_{inc} en utilisant chaque vecteur Pre_i :

$$\begin{aligned} C_{Pre1} &= [01\text{-}10100000000000000000000]^T \\ C_{Pre2} &= [010010000000000\text{-}10000000000]^T \\ C_{Pre3} &= [010010000000000\text{-}1000000000]^T \\ C_{Pre11} &= [011010000000000000000\text{-}100000]^T \\ C_{Pre128} &= [011010000000000\text{-}1\text{-}1000\text{-}100\text{-}1\text{-}1\text{-}1]^T \end{aligned}$$

5. Déterminer l'ensemble des vecteurs C_{poss} qui conservent chaque invariant de place du RdP original, cet ensemble est noté EI_r.

$$\begin{aligned} EI_1 &= C_{Pre1} \\ EI_2 &= C_{Pre1} \\ EI_3 &= C_{Pre1} \\ EI_4 &= C_{Pre1} \\ EI_5 &= C_{Pre1} \\ EI_6 &= C_{Pre1} \\ EI_7 &= C_{Pre1} \\ EI_8 &= C_{Pre1} \end{aligned}$$

6. Déterminer E_{sol}, l'ensemble de tous les vecteurs C_{Prei} qui conservent tous les invariants de places du réseau de Petri original.

$$E_{sol} = C_{Pre1}$$

Dans l'ensemble de solutions présenté C_{Pre1} constitue la seule modification qui conserve tous les invariants de places du réseau. Nous allons montrer la modification de MOD_C basée sur C_{Pre1}.

7. Échanger l'ancien vecteur colonne de t_{inc} avec le vecteur contenu dans E_{sol} pour former la matrice modifiée notée C_{RC} qui rétablit la cohérence. Cette matrice est donnée figure 7.13 :
8. Construire le nouveau RdP du système défini par la matrice d'incidence C_{RC}. Ce réseau est montré dans la figure 7.14

   Avec la méthode de diagnostic, nous avons isolé du modèle la partie qui bloquait le franchissement de t2(a2.a4), cette partie correspond à la place p4 et la transition t1(b2). Désormais la transition t2(a2.a4) est franchissable à partir du marquage courant du réseau. Cependant, cette transition est en conflit avec la transition

C_{RC} =

t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	p_1
-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_2
1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_3
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_4
0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_5
0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_6
0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_7
0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_8
0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	p_9
0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	p_{10}
0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	p_{11}
0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	p_{12}
0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	p_{13}
0	0	0	0	0	0	0	0	0	0	0	1	-1	-1	0	0	-1	0	0	0	0	p_{14}
0	0	0	0	0	0	0	0	0	0	0	0	1	0	-1	0	0	0	0	0	0	p_{15}
0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0	0	p_{16}
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0	p_{17}
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	-1	0	0	0	0	0	p_{18}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	p_{19}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0	p_{20}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	p_{21}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	p_{22}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1	p_{23}
0	0	0	-1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	p_{24}
0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	p_{25}
0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	1	1	p_{26}

FIG. 7.13 – Matrice d'incidence modifiée pour la commande C_{RC}

t1(b2), le franchissement de cette dernière transition amène le réseau vers un marquage mort.

Caractérisation de l'observation incohérente

La partie isolée du modèle peut représenter soit une défaillance dans le système ou soit une erreur de modélisation dans ce modèle. Nous pouvons distinguer ces deux suppositions à travers les suppositions réalisées sur la cohérence de cette observation par rapport à MOD_{CR}. Nous présentons ensuite ces deux conclusions et la relation avec MOD_{CR} :

1. La supposition de la cohérence de l'observation avec MOD_{CR} nous permet de conclure que l'observation correspond à un comportement normal et la partie isolée, dans le modèle de la commande, correspond à une erreur du modèle. A l'élimination de cette partie le système retrouve à nouveau les bonnes propriétés du modèle original. Or, il serait possible qu'après l'élimination de ce comportement nous puissions observer un comportement correspondant à cette partie du modèle. Cette partie est donc conservée dans le modèle.
2. La supposition de l'incohérence de l'observation avec MOD_{CR} nous permet de conclure que l'observation est causée par un dysfonctionnement du système et non par un comportement normal. La partie isolée dans le modèle de la commande pourrait représenter le comportement normal ou non mais en tout cas maintenant cette partie représente le changement de fonctionnement du système.

La prochaine observation que nous allons présenter correspond à une observation qui nous fournit une connaissance encore différente des précédentes. Pour cela nous allons

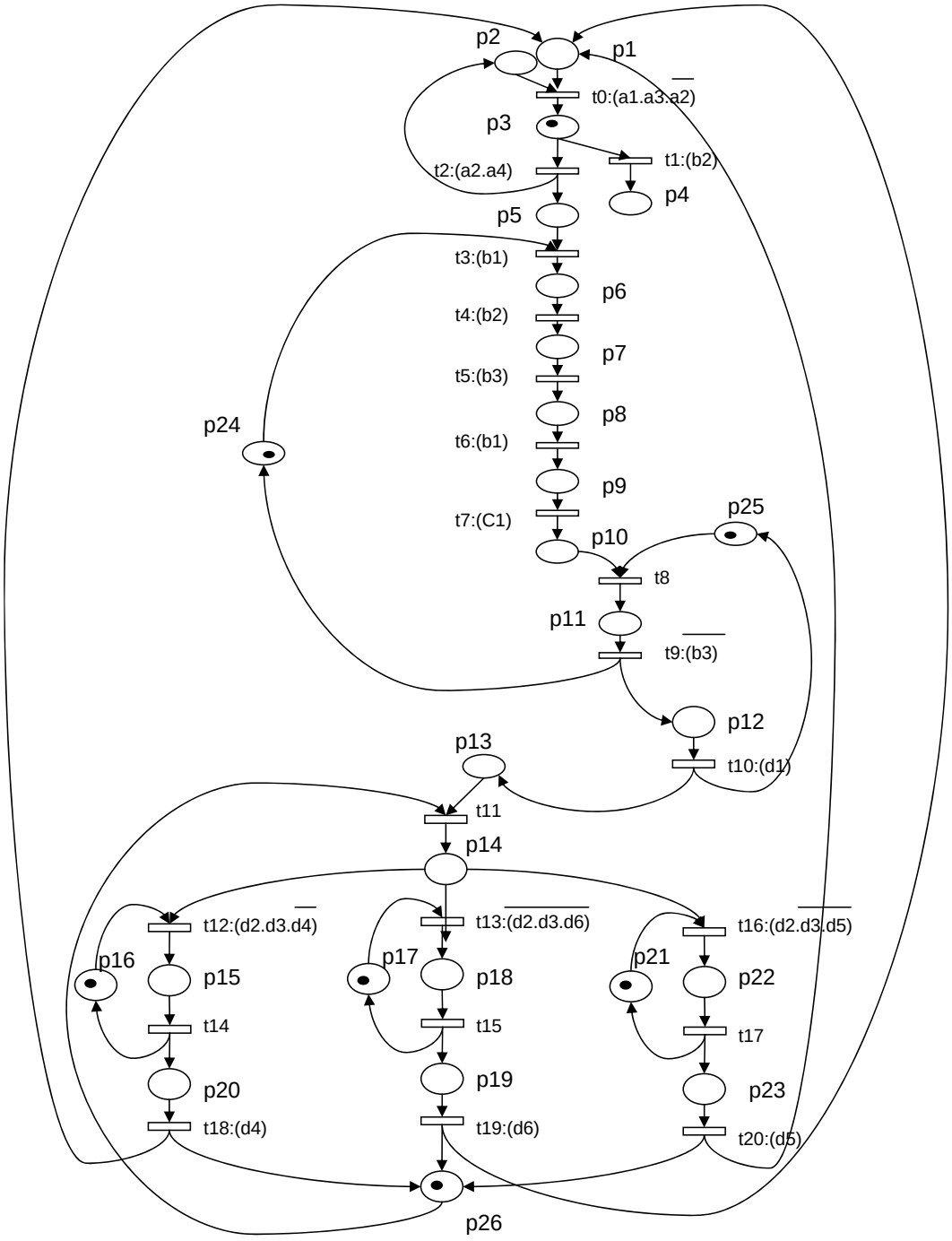


FIG. 7.14 – RdP de MOD_C modifié

supposer que le procédé a évolué de façon normale et que la dernière observation de la séquence observée $S_{ev} = (a1.a3), (a2.a4), (b1), (b2), (b3), (b1)$ amène le système vers l'état M_6 du MOD_{CA} (figure 23, page 161) et dans l'état M_7 du MOD_C (figure 27 page 164). Nous allons présenter la suite de cette séquence observée.

7.3.3 Analyse de la troisième observation

7.3.3.1 Modélisation du comportement observé avec MOD_{OBS}

Nous allons supposer qu'après la séquence d'observation correspondant à la séquence d'événements $S_{ev} = (a_3.a_1), (a_4.a_2), (b_1), (b_2), (b_3), (b_1)$, une troisième observation associée à l'événement (c_1). La phase de détection est initiée avec l'intégration de cette observation au modèle d'observations. Le nouvel MOD_{OBS} est représenté graphiquement par la figure 7.15. Pour cette intégration nous procédons comme suit :

prise en compte de la possibilité d'évolution P_1

$$M_6^{OBS} \xrightarrow{t7(c1)} M_6^{OBS}$$

prise en compte des possibilités d'évolution P_2

$$M_6^{OBS} \xrightarrow{t7(c1)} M_5^{OBS}$$

$$M_6^{OBS} \xrightarrow{t7(c1)} M_4^{OBS}$$

$$M_6^{OBS} \xrightarrow{t7(c1)} M_3^{OBS}$$

$$M_6^{OBS} \xrightarrow{t7(c1)} M_2^{OBS}$$

$$M_6^{OBS} \xrightarrow{t7(c1)} M_1^{OBS}$$

$$M_6^{OBS} \xrightarrow{t7(c1)} M_0^{OBS}$$

prise en compte des possibilités d'évolution P_3

$$M_6^{OBS} \xrightarrow{t7(c1)} M_x^{OBS}$$

Pour représenter les possibilités P_2 six transitions sont utilisées, chacune de ces transitions part de l'état courant M_6 et arrive à un état différent déjà existant dans le modèle d'observations. Au total nous utilisons huit transitions pour représenter les trois possibilités. Nous allons chercher à vérifier si une de ces possibilités est vraie.

Nous allons passer maintenant à l'étape de vérification de la cohérence pour la dernière observation. Nous commençons par vérifier cette observation avec MOD_{CA}.

7.3.3.2 La détection par vérification de cohérence

La vérification de la cohérence entre MOD_{OBS} **et** MOD_{CA}

La figure 7.16 montre la représentation partielle de $GVER^{CA}$ utilisé pour déterminer la cohérence entre l'observation et MOD_{CA}. $GVER^{CA}$ est composé de 96 états et 90 événements. En analysant chaque possibilité pour la vérification nous trouvons les résultats suivants :

P_1 :

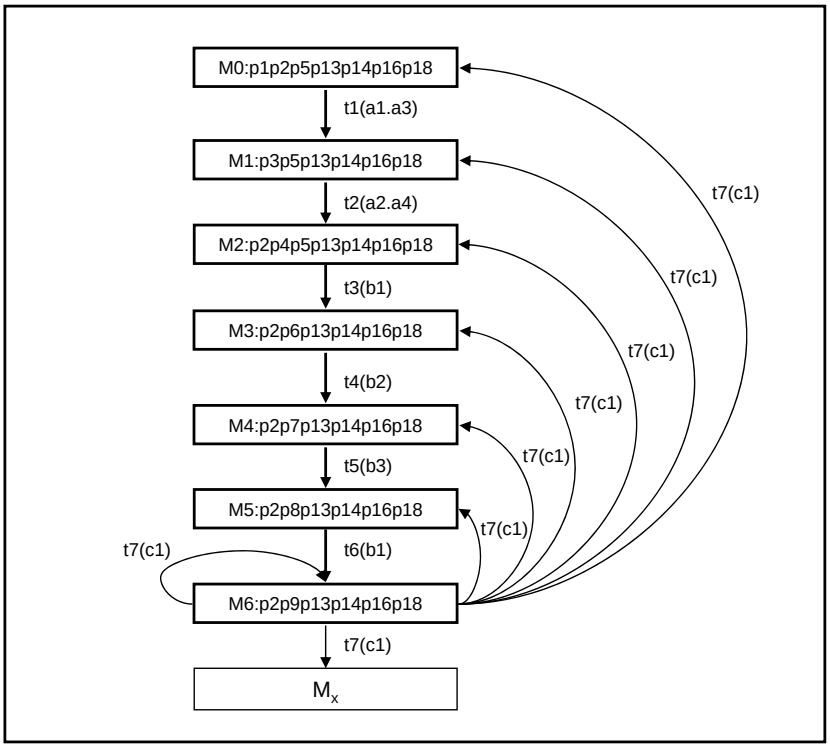


FIG. 7.15 – MOD_{OBS} après plusieurs évolutions

$$1.\ T_{Rd1} = M_6^{OBS}, M_6^1 \xrightarrow{(c1)^{OBS}} M_6^{OBS}, M_6^1$$

Étant donné que seul MOD_{OBS} est réceptif à l'événement observé à partir de l'état courant et que MOD_{CA} n'est pas sensible à cet événement dans son état courant, alors ces comportements sont incohérents. Nous vérifions maintenant la deuxième possibilité de MOD_{OBS}.

P_2 :

$$2.\ T_{Rd2} = M_6^{OBS}, M_6^1 \xrightarrow{(c1)^{OBS}} M_5^{OBS}, M_6^1$$

$$3.\ T_{Rd3} = M_6^{OBS}, M_6^1 \xrightarrow{(c1)^{OBS}} M_4^{OBS}, M_6^1$$

$$4.\ T_{Rd4} = M_6^{OBS}, M_6^1 \xrightarrow{(c1)^{OBS}} M_3^{OBS}, M_6^1$$

$$5.\ T_{Rd5} = M_6^{OBS}, M_6^1 \xrightarrow{(c1)^{OBS}} M_2^{OBS}, M_6^1$$

$$6.\ T_{Rd6} = M_6^{OBS}, M_6^1 \xrightarrow{(c1)^{OBS}} M_1^{OBS}, M_6^1$$

$$7.\ T_{Rd7} = M_6^{OBS}, M_6^1 \xrightarrow{(c1)^{OBS}} M_0^{OBS}, M_6^1$$

Dans cette possibilité nous ne retrouvons pas de comportements synchrones car M_6^1 n'est pas sensible à l'événement (c_1). Avec cet événement, le seul modèle qui évolue est MOD^{OBS}. Nous vérifions alors la troisième possibilité.

P_3 :

$$8.\ T_{Rd8} = M_6^{OBS}, M_6^1 \xrightarrow{(c1)^{OBS}} M_x^{OBS}, M_6^1$$

Cette possibilité n'est pas vérifiée car comme pour P_1 et P_2, le modèle de comportement attendu n'est pas sensible à (c_1) dans son état courant. Cette observation est donc incohérente avec le modèle de comportement attendu.

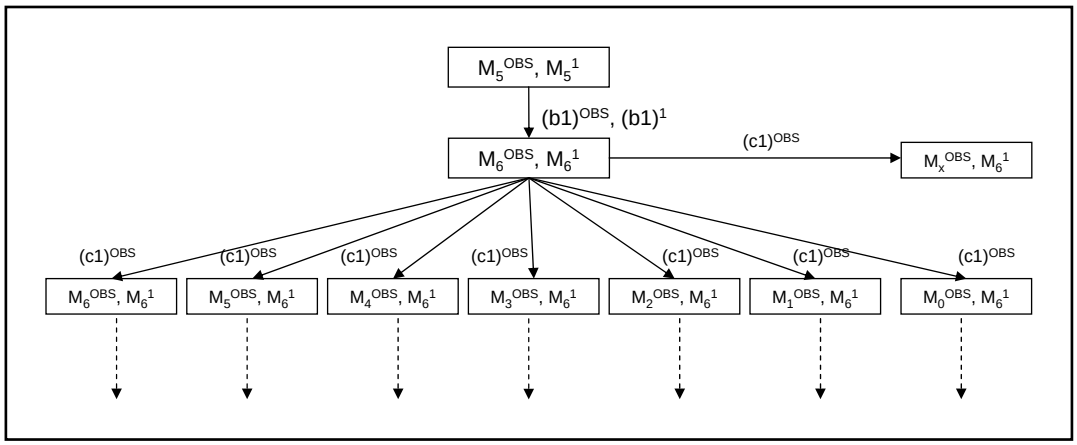


FIG. 7.16 – $GVER^{CA}$

Nous allons passer maintenant à la vérification de la cohérence entre l'observation et le modèle de la commande.

Vérification de la cohérence entre MOD_{OBS} et MOD_C

La figure 7.17 montre une représentation partielle de $GVER^C$ utilisé pour déterminer la cohérence entre l'observation et MOD_C. Ce graphe possède 152 états et 111 événements. Nous allons étudier les possibilités pour déterminer la cohérence :

P_1 :

$$1.\ T_{Rf1} = M_6^{OBS}, M_7^2 \xrightarrow{(c1)^{OBS},(c1)^2} M_6^{OBS}, M_8^2$$

MOD_C a évolué vers un état différent de l'état décrit par P_1, alors ces comportements ne sont pas cohérents. Nous vérifions la deuxième possibilité.

P_2 :

$$2.\ T_{Rf2} = M_6^{OBS}, M_7^2 \xrightarrow{(c1)^{OBS},(c1)^2} M_5^{OBS}, M_8^2$$

$$3.\ T_{Rf3} = M_6^{OBS}, M_7^2 \xrightarrow{(c1)^{OBS},(c1)^2} M_4^{OBS}, M_8^2$$

$$4.\ T_{Rf4} = M_6^{OBS}, M_7^2 \xrightarrow{(c1)^{OBS},(c1)^2} M_3^{OBS}, M_8^2$$

$$5.\ T_{Rf5} = M_6^{OBS}, M_7^2 \xrightarrow{(c1)^{OBS},(c1)^2} M_2^{OBS}, M_8^2$$

$$6.\ T_{Rf6} = M_6^{OBS}, M_7^2 \xrightarrow{(c1)^{OBS},(c1)^2} M_1^{OBS}, M_8^2$$

$$7.\ T_{Rf7} = M_6^{OBS}, M_7^2 \xrightarrow{(c1)^{OBS},(c1)^2} M_0^{OBS}, M_8^2$$

A partir de l'état courant M_6^{OBS} MOD_{OBS} peut évoluer avec l'événement (c_1) vers des états différents M_0^{OBS}, M_1^{OBS}, M_2^{OBS}, M_3^{OBS}, M_4^{OBS}, M_5^{OBS}. A partir de M_7^2 MOD_C peut évoluer uniquement vers l'état M_8^2, étant donné que M_8^2 n'est pas un état avec les caractéristiques décrites par la possibilité, ces comportements ne sont pas synchrones. Cette possibilité n'est pas vraie, nous passons alors à la troisième possibilité.

P_3 :

$$8.\ T_{Rf8} = M_6^{OBS}, M_7^2 \xrightarrow{(c1)^{OBS},(c1)^2} M_x^{OBS}, M_8^2$$

L'état de M_8^2 atteint pour l'observation correspond bien aux caractéristiques requises par P_3, car c'est un état qui n'a pas été précédemment parcouru. La cohérence entre le comportement observé et le comportement décrit dans la commande est donc établit.

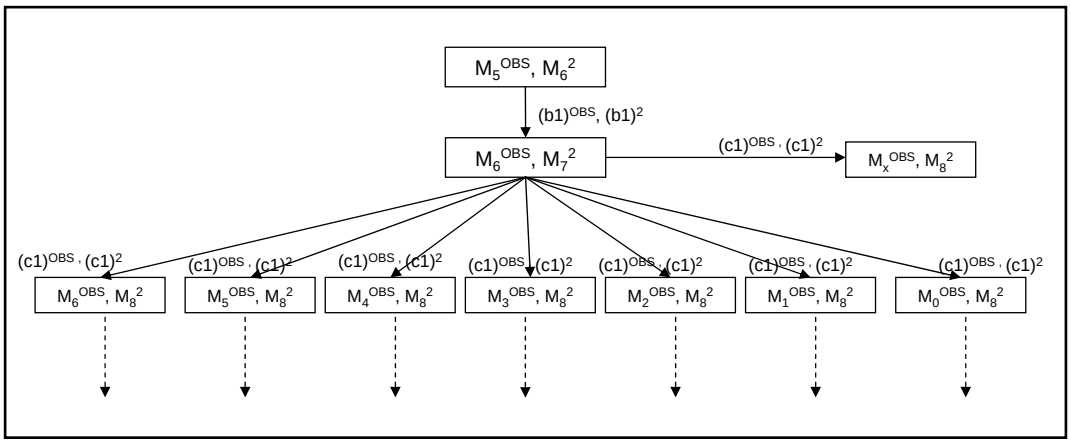


FIG. 7.17 – $GVER^C$

Cependant M_x^{OBS} ne peut pas prendre la valeur de M_8^2 car le modèle d'observation représente le comportement observé compatible avec celui décrit par MOD_{CA}, tandis que MOD_C représente la commande appliquée au procédé. M_x^{OBS} devra prendre la valeur de MOD_{CA}. Pour déterminer la valeur de M_x^{OBS} nous devons d'abord réaliser la modification de MOD_{CA} pour le rendre cohérent avec ce comportement observé et puis trouver la valeur de M_x^{OBS}. Nous savons maintenant que la modification à réaliser à MOD_{CA} doit permettre d'atteindre un marquage avec les caractéristiques de M_x^{OBS}, c'est-à-dire, ce marquage doit appartenir à l'ensemble des marquages non visités M_{NF} de MOD_{CA}.

7.3.3.3 Identification de la configuration

Avec cette dernière observation nous obtenons des informations fournies par l'étape de la vérification de cohérence entre cette observation et les modèles du système. L'observation est incohérente avec MOD_{CA} et est cohérente avec MOD_C. La valeur de cette constatation peut être complétée avec les suppositions nécessaires pour MOD_{CR} :

1. cohérence avec $MOD_{CR} : CNS_{ARC}(T_{Rf8}) = (0, 1, 1)$
2. incohérente avec $MOD_{CR} : CNS_{ARC}(T_{Rf8}) = (0, 0, 1)$

Ces constatations nous apportent la connaissance de l'existence de deux nouveaux sous-ensembles dans notre configuration. Si nous prenons la constatation 1, la connaissance déduit de cette constatation est l'existence du sous-ensemble type CR. La valeur de la deuxième constatation, la connaissance apportée indique l'existence du sous-ensemble type C.

Nous allons passer maintenant à l'étape de diagnostic pour rendre MOD_{CA} cohérent avec l'observation c_1.

7.3.3.4 Le diagnostic

Nous allons modifier MOD_{CA} en utilisant le cas P_3 du MOD_{OBS} pour réaliser cette modification (T_{Rf8}) en rendant franchissable la transition associée à l'événement observé à partir du marquage courant M_6 de MOD_{CA}. Nous appliquons la démarche de diagnostic pour cette modification en montrant uniquement les résultats les plus importants.

1. Détermination de l'ensemble des places marquées de MOD_{CA} dans l'état courant M_6^2 :

$$PM_k = \{p_2, p_9, p_{13}, p_{14}, p_{16}, p_{18}\}$$

2. Calculer l'ensemble de toutes les nouvelles places d'entrée possibles à la transition incohérente t_{inc}, où $t_{inc} = t7$. Au total nous avons 2^6 soit 64 possibilités. Nous présentons ici les résultats qui nous aident à la modification.

   M_{poss} = {$\emptyset$, {p_2}, {p_9}, {p_{13}}, {p_{14}}, {p_{16}}, {p_{18}}, {p_2, p_9}, {p_2, p_{13}}, {p_2, p_{14}}, {p_2, p_{16}}, {p_2, p_{18}}, {p_9, p_{13}},.....{p_2, p_9, p_{13}, p_{14}, p_{16}, p_{18}}

3. Construire l'ensemble de tous les vecteurs de places d'entrée possibles Pre_{poss} à la transition incohérente $t_{inc} = t7$ en utilisant les sous-ensembles de places de M_{poss}. Étant donné la taille des vecteurs nous allons les présenter leur forme transposée :

$$\begin{aligned}
Pre_1 &= [0100000000000000]^T \\
Pre_2 &= [0000000010000000]^T \\
Pre_3 &= [0000000000100000]^T \\
Pre_4 &= [0000000000010000]^T \\
Pre_7 &= [0100000010000000]^T \\
Pre_8 &= [0100000000100000]^T \\
Pre_{12} &= [0000000010001000]^T \\
Pre_{64} &= [0100000010001101]^T
\end{aligned}$$

4. Composer les nouvelles colonnes de la matrice d'incidence C_{poss} pour la transition incohérente t_{inc}, en utilisant chaque vecteur Pre_i :

$$\begin{aligned}
C_{Pre1} &= [0\text{-}10010000010000000]^T \\
C_{Pre2} &= [00001000\text{-}1010000000]^T \\
C_{Pre3} &= [000010000010\text{-}100000]^T \\
C_{Pre4} &= [0000100000100\text{-}10000]^T \\
C_{Pre7} &= [0\text{-}1001000\text{-}1010000000]^T \\
C_{Pre8} &= [0\text{-}10010000010\text{-}100000]^T \\
C_{Pre12} &= [00001000\text{-}1000\text{-}100000]^T \\
C_{Pre64} &= [0\text{-}1001000\text{-}1010\text{-}1\text{-}10\text{-}10\text{-}1]^T
\end{aligned}$$

5. Déterminer l'ensemble des vecteurs C_{Prei} qui conservent chaque invariant de place du RdP original. Cet ensemble est noté EI_r.

$EI_1 = \{C_{Pre2}, C_{Pre7}, C_{Pre8}, C_{Pre12}, C_{Pre64}\}$

$EI_2 = \{C_{Pre2}, C_{Pre7}, C_{Pre12}, C_{Pre64}\}$

$EI_3 = \{C_{Pre1}, C_{Pre2}, C_{Pre3}, C_{Pre4}, C_{Pre7}, C_{Pre8}, C_{Pre12}\}$

$EI_4 = \{C_{Pre2}, C_{Pre3}, C_{Pre4}, C_{Pre12}\}$

$EI_5 = \{C_{Pre1}, C_{Pre2}, C_{Pre3}, C_{Pre4}, C_{Pre7}, C_{Pre8}, C_{Pre12}\}$

$EI_6 = \{C_{Pre1}, C_{Pre2}, C_{Pre3}, C_{Pre7}, C_{Pre8}, C_{Pre12}\}$

$EI_7 = \{C_{Pre3}, C_{Pre8}, C_{Pre12}, C_{Pre64}\}$

6. Déterminer E_{sol}, l'ensemble de tous les vecteurs C_{Prei} qui conservent tous les invariants de places du réseau de Petri original.

$Esol = \{C_{Pre12}\}$

C_{Pre12} est la seule modification qui conserve tous les invariants de place du réseau. Nous allons montrer la modification du MOD_{CA} basée sur la nouvelle définition C_{Pre12} de la transition t_7.

7. Échanger l'ancien vecteur colonne de t_{inc} pour le vecteur contenu dans E_{sol} pour former la matrice modifiée qui rétablit la cohérence notée C_{RC}. La figure 7.18 montre cette nouvelle matrice :

8. Construire le nouveau réseau de Petri du système défini par la matrice d'incidence C_{RC}. Ce réseau modifié est donné dans la figure 7.19 :

7.3.4 Identification de la configuration par le graphe des configurations

Nous allons utiliser la connaissance apportée par les observations pour identifier la configuration des trois modèles du système.

Le cumul des connaissances successivement potentiellement apportées par les observations est déjà compilé dans le graphe des configurations décrit au chapitre 5. En ligne,

C_{RC} =

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	
	-1	0	0	0	0	0	0	0	0	0	0	1	1	1	p_1
	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	p_2
	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	p_3
	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	p_4
	0	0	-1	0	0	0	0	1	0	0	0	0	0	0	p_5
	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	p_6
	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	p_7
	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	p_8
	0	0	0	0	0	1	-1	-1	0	0	0	0	0	0	p_9
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	p_{10}
	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	p_{11}
	0	0	0	0	0	0	0	0	1	-1	-1	0	-1	0	p_{12}
	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	p_{13}
	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	p_{14}
	0	0	0	0	0	0	0	0	0	1	0	-1	0	0	p_{15}
	0	0	0	0	0	0	0	0	0	0	-1	0	0	1	p_{16}
	0	0	0	0	0	0	0	0	0	0	1	0	0	-1	p_{17}
	0	0	0	0	0	0	0	0	-1	0	0	1	1	1	p_{18}

FIG. 7.18 – La matrice d'incidence C_{RC}

à chaque observation, ou après une séquence d'observation, il est possible de parcourir le graphe pour déterminer les configurations possibles des trois ensembles.

Première observation

L'observation correspondant à l'événement $(a_1.a_3)$ nous a permis d'obtenir les valeurs des constatations suivantes :

$$1. - CNS_{ARC}(T_{Rb}) = (1,1,1) \text{ (cohérence avec } MOD_{CR})$$
$$2. - CNS_{ARC}(T_{Rb}) = (1,0,1) \text{ (incohérence avec } MOD_{CR})$$

La connaissance K_{ARC} apportée pour chaque constatation correspond à la déduction du sous-ensemble contenu dans la configuration recherchée.

$$1. - K_{ARC} = \langle CNS_{ARC}(T_{Rb}) = (1,1,1)\rangle = \langle ARC\rangle$$
$$2. - K_{ARC} = \langle CNS_{ARC}(T_{Rb}) = (1,0,1)\rangle = \langle AC\rangle$$

Nous obtenons deux séquences de constatations TC_i avec cette observation. Nous montrons ensuite chaque TC_1 et TC_2 ainsi que le support correspondant à chaque séquence de constatations.

$$1. - TC_1 = \langle(1,1,1)\rangle \text{ avec le support } S(TC_1) = \{T_{Rb}\}$$
$$2. - TC_2 = \langle(1,0,1)\rangle \text{ avec le support } S(TC_2) = \{T_{Rb}\}$$

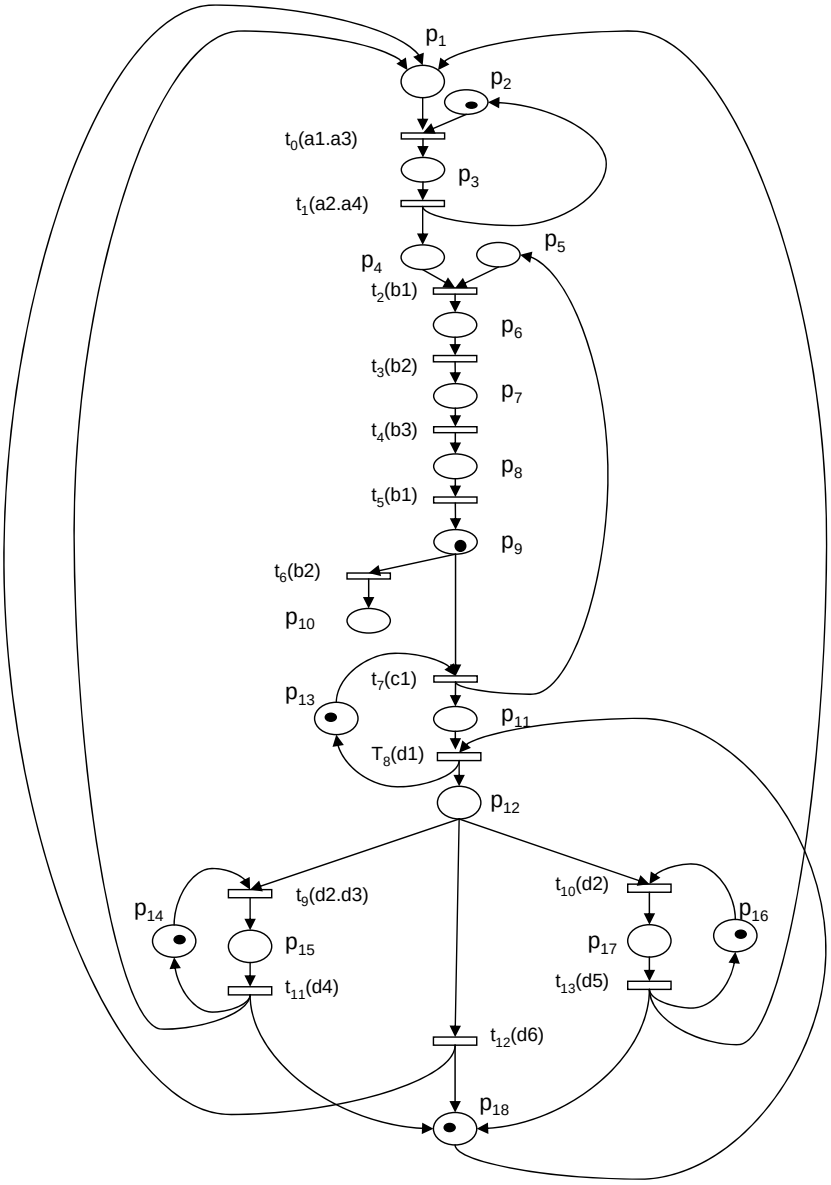


FIG. 7.19 – Le RdP_{CA} modifié

Les configurations qui correspondent à chaque séquence de constatations TC_i réalisées jusqu'à la première observation sont les suivantes. Les numéros de configurations font référence au tableau 28 décrit dans l'annexe à la page 165 :

$TC_1 \rightarrow Ensemble - solution =$ {1 , 3 , 5 , 7 , 10 , 12 , 13 , 17 , 20 , 23 , 25 , 26 , 32 , 34 , 35 , 38 , 40 , 41 , 43 , 44 , 45 , 49 , 52 , 54 , 55 , 58 , 60 , 61 , 63 , 64 , 65 , 68 , 70 , 71 , 73 , 74 , 75 , 77 , 78 , 79 , 80 , 81 , 83 , 84 , 85 , 86 , 87 , 90 , 92 , 93 , 95 , 96 , 97 , 99 , 100 , 101 , 103 , 104 , 105 , 106 , 107 , 108 , 109}

$TC_2 \rightarrow Ensemble - solution =$ { 4 , 8 , 11 , 12 , 18 , 21 , 22 , 23 , 27 , 30 , 31 , 32 , 36 , 37 , 38 , 42 , 43 , 44 , 45 , 50 , 51 , 52 , 56 , 57 , 58 , 62 , 63 , 64 , 66 , 67 , 68 , 72 , 73 , 74 , 76 , 77 , 78 , 80 , 81 , 82 , 83 , 84 , 86 , 87 , 88 , 89 , 90 , 94 , 95 , 96 , 98 , 99 , 100 , 102 , 103 , 104 , 106, 107, 108, 109 }

Dans la première séquence de constatations, nous avons 63 configurations acceptables. Pour la deuxième séquence de constatations, 60 configurations sont compatibles avec la constatation contenue dans TC_2.

Maintenant nous allons analyser la deuxième observation et montrer comment l'ensemble-solution de chaque séquence de constatations est réduit.

Observation o2

L'observation correspondant à la trajectoire T_{Rc3} est cohérente avec MOD_{CA} et incohérente avec MOD_C. Ceci donne les constatations suivantes. :

$1. - CNS_{ARC}(T_{Rc3}) = (1,1,0)$ (cohérence avec MOD_{CR}) $2. - CNS_{ARC}(T_{Rc3}) = (1,0,0)$ (incohérence avec MOD_{CR})

La connaissance K_{ARC} apportée par les constatations correspondantes est :

$$1. - K_{ARC} = (1,1,0)\rangle = \langle AR\rangle$$
$$2. - K_{ARC} = (1,0,0)\rangle = \langle A\rangle$$

En cumulant ces nouvelles connaissances à celles apportées par la première observation nous obtenons quatre séquences de constatations possibles.

$1. - TC_1 = \langle(1,1,1),(1,1,0)\rangle \mid K_{ARC}(TC_1) = \langle ARC, AR\rangle$

$2. - TC_2 = \langle(1,1,1),(1,0,0)\rangle \mid K_{ARC}(TC_2) = \langle ARC, A\rangle$

$3. - TC_3 = \langle(1,0,1),(1,1,0)\rangle \mid K_{ARC}(TC_3) = \langle AC, AR\rangle$

$4. - TC_4 = \langle(1,0,1),(1,0,0)\rangle \mid K_{ARC}(TC_4) = \langle AC, A\rangle$

Les configurations qui correspondent à ces quatre séquences de constatations sont

TC_1 : {10, 25, 34, 40, 43, 45, 54, 60, 63, 65, 70, 73, 75, 77, 79, 80, 81, 83, 85, 86, 92, 95, 97, 99, 101, 103, 105, 106, 107, 108, 109}

TC_2 : {3, 17, 20, 23, 25, 26, 49, 52, 54, 55, 58, 60, 61, 63, 64, 65, 81, 90, 92, 93, 95, 96, 97, 99, 100, 101, 103, 104, 105, 106, 107, 109}

TC_3 : {23, 42, 43, 62, 66, 72, 73, 76, 77, 80, 81, 82, 83, 86, 87, 88, 94, 95, 98, 99, 102, 103, 106, 107, 108, 109}

TC_4 : {18, 21, 22, 23, 50, 51, 52, 56, 57, 58, 62, 63, 64, 81, 88, 89, 90, 94, 95, 96, 98, 99, 100, 102, 103, 104, 106, 107, 109}

L'ensemble-solution qui correspond aux constatations de TC_1 est composé par 31 configurations, l'ensemble-solution de TC_2 a été réduit à 32 configurations seulement. Pour la TC_3 cet ensemble est réduit à 26 configurations, finalement pour TC_4 son ensemble-solution est composé par 29 configurations.

Maintenant, nous allons présenter la dernière observation qui peut nous aider à identifier la ou les configurations qui correspondent à ces trois observations.

Observation o7

La prise en compte de la troisième observation va nous permettre de préciser encore la configuration des modèles.

$$1. - CNS_{ARC}(T_{Rf8}) = (0,1,1) \text{ (cohérence avec } MOD_{CR})$$
$$2. - CNS_{ARC}(T_{Rf8}) = (0,0,1) \text{ (incohérence avec } MOD_{CR})$$

La connaissance K_{ARC} apparaissent :

$$1. - K_{ARC} = \langle (0,1,1) \rangle = \langle RC \rangle$$
$$2. - K_{ARC} = \langle (0,0,1) \rangle = \langle C \rangle$$

Avec ces trois observations o1, o2 et o7 vont produire au total huit séquences de constatations. Ces huit séquences de constatations et les connaissances associées sont les suivantes :

$1. - TC_1 = \langle (1,1,1),(1,1,0),(0,1,1) \rangle \mid K_{ARC}(TC_1) = \langle ARC, AR, RC \rangle$

$2. - TC_2 = \langle (1,1,1),(1,1,0),(0,0,1) \rangle \mid K_{ARC}(TC_2) = \langle ARC, AR, C \rangle$

$3. - TC_3 = \langle (1,1,1),(1,0,0),(0,1,1) \rangle \mid K_{ARC}(TC_3) = \langle ARC, A, RC \rangle$

$4. - TC_4 = \langle (1,1,1),(1,0,0),(0,0,1) \rangle \mid K_{ARC}(TC_4) = \langle ARC, A, C \rangle$

$5. - TC_5 = \langle (1,0,1),(1,1,0),(0,1,1) \rangle \mid K_{ARC}(TC_5) = \langle AC, AR, RC \rangle$

$6. - TC_6 = \langle (1,0,1),(1,1,0),(0,0,1) \rangle \mid K_{ARC}(TC_6) = \langle AC, AR, C \rangle$

$7. - TC_7 = \langle (1,0,1)),(1,0,0)),(0,1,1)) \rangle \mid K_{ARC}(TC_7) = \langle AC, A, RC \rangle$

$8. - TC_8 = (1,0,1)),(1,0,0)),(0,0,1)) \rangle \mid K_{ARC}(TC_8) = \langle AC, A, C \rangle$

Les configurations correspondant à ces séquences de constatations sont les suivantes :

TC_1 : {45, 65, 75, 79, 80, 81, 85, 86, 97, 101, 105, 106, 107, 108, 109}

TC_2 : {34, 60, 70, 77, 79, 80, 85, 92, 99, 101, 103, 107, 108, 109}

TC_3 : {26, 55, 61, 64, 65, 81, 93, 96, 97, 100, 101, 104, 105, 106, 107, 109}

TC_4 : {17, 49, 58, 60, 61, 90, 92, 93, 99, 100, 101, 103, 104, 105, 107, 109}

TC_5 : {62, 72, 76, 80, 81, 82, 83, 86, 87, 88, 94, 98, 102, 106, 107, 108, 109}

TC_6 : {66, 76, 77, 82, 83, 87, 88, 98, 99, 102, 103, 107, 108, 109}

TC_7 : {57, 64, 81, 89, 94, 96, 98, 100, 102, 104, 106, 107, 109}

TC_8 : {56, 57, 58, 88, 89, 90, 98, 99, 100, 102, 103, 104, 107, 109}

Le parcours du graphe d'identification des configurations à chaque observation aboutit dans les ensembles-solutions qui contiennent des ensembles des configurations. Ces configurations varient selon les valeurs de chaque constatation contenue dans une séquence de constatations. L'identification des configurations des modèles fait progresser la connaissance globale dont dispose le système de commande, de surveillance et de reconfiguration. L'utilisation de ces résultats est aujourd'hui l'objet de nos perspectives.

Signalons que l'analyse exhaustive des différentes configurations a été faites en considérant, à chaque observation, les conséquences de la cohérence avec MOC_{CR} ou de l'incohérence avec ce même modèle. Dans une application réelle, une seule trajectoire serait suivie. En effet, chaque observation, est cohérente ou non avec MOD_{CR}.

Conclusions

Dans ce chapitre nous avons appliqué complètement notre approche de détection et diagnostic à base de cohérence. Le système présenté ici est une maquette de tri visible à l'INSA Toulouse. Nous avons montré les modèles de comportement attendu MOD_{CA} et le modèle de la commande MOD_C pour le tri d'une pièce avec le but de bien montrer notre approche. Trois observations issues du fonctionnement du procédé ont été analysées dans le chapitre car ce sont ces trois observations qui nous permettent de montrer le résultat obtenu avec l'application de notre approche.

Pour chaque observation nous avons montré la construction du modèle d'observation MOD_{OBS}. Ce modèle a été utilisé pour la phase de vérification de la cohérence entre le comportement observé et le comportement décrit dans les modèles du système. Le produit cartésien entre les modèles est implémenté pour cette phase de vérification. La détermination de la cohérence dans cette phase est réalisée par la synchronisation des comportements entre MOD_{OBS} et MOD_{CA}, et entre MOD_{OBS} et MOD_C. Si la synchronisation n'est pas possible entre MOD_{OBS} et un des modèles, alors l'observation en question est déclarée incohérente avec le modèle respectif et dans ce cas là une incohérence est détectée.

A la détection d'une incohérence, la fonction de diagnostic se charge de rétablir la cohérence entre l'observation et le modèle en question. Dans cet exemple la deuxième observation a conduit à la modification de MOD_C.

La troisième observation présente le cas dual de la deuxième et débouche sur la modification de MOD_{CA}.

A chaque observation, nous avons fait deux suppositions par rapport à la cohérence entre l'observation et le modèle de comportement réel. Chacune de ces observations apporte des connaissances qui nous ont permis de parcourir le graphe d'identification des configurations et d'identifier aussi finement que le permettent les observations, la configuration réelle des ensembles de trajectoires des modèles utilisés dans notre approche.

Conclusions et perspectives

Cette thèse traite le problème de détection et diagnostic de défaillances dans les systèmes à événements discrets.

Notre approche est une approche basée cohérence, qui utilise des modèles représentant uniquement le comportement normal du système. Les modèles utilisés pour la détection des symptômes de défaillances sont au centre de ce travail. Nous considérons que ces modèles sont susceptibles de contenir des erreurs de modélisation. L'origine de ces erreurs peut être due aux aléas propres à la modélisation ou à la méconnaissance du fonctionnement du système surveillé. Quelle que soit la cause de ces erreurs, il est difficile de distinguer leurs manifestations (faux symptômes) et les symptômes causés par le dysfonctionnement du système (symptômes de défaillance).

La méthode que nous avons proposée réalise la détection et le diagnostic en prenant en compte la possibilité d'erreurs dans les modèles du système servant de référence du comportement correct : un modèle du comportement attendu et un modèle de la commande mis en place par réseau de Petri. Nous avons mis en évidence que ces erreurs de modélisation produisent des écarts que nous appelons configuration des ensembles de trajectoires des modèles du système.

Les deux modèles (des comportements attendus et de la commande) sont confrontés avec le comportement observé pour détecter des incohérences. Dans certains cas seul un des deux modèles peut être incohérent. Le problème qui se pose alors est que ce résultat peut provenir d'une erreur de modélisation comme d'une défaillance du procédé. Pour distinguer ces deux cas, il est nécessaire d'introduire un troisième modèle qui est le modèle du comportement réel du système. Bien que l'utilisation du modèle du comportement réel augmente le nombre de configurations entre les modèles, elle permet, d'une part, de distinguer les vrais symptômes des incohérences dues aux erreurs du modèle et, d'autre part de conclure sur le modèle incorrect.

Nous avons montré qu'une séquence d'observations permet d'identifier un ensemble de configurations. Lorsque cette séquence révèle l'existence des sept ensembles relatifs à trois modèles, la configuration peut être identifiée sans ambiguïté. C'est d'ailleurs la seule configuration qui peut être déterminée avec certitude. Lors du diagnostic, il est ainsi possible d'appliquer les modifications nécessaires aux niveaux du ou des modèles incriminés pour rétablir la cohérence.

Les suites de ce travail sont nombreuses. A court terme il serait intéressant de compléter les résultats obtenus par l'étape d'identification par des d'informations supplémentaires pour pouvoir à coup sûr faire la distinction entre les deux types d'incohérence. Dans le cas où un faux symptôme serait confirmé il serait ainsi possible d'éliminer avec certitude une erreur dans le modèle lors de l'étape de diagnostic avec la garantie de ne pas intégrer un comportement défaillant au modèle.

Au niveau de la détection, nous envisageons également d'exploiter l'ensemble des contraintes et des propriétés dictées par le cahier des charges. En effet, la confrontation du comportement observé avec le comportement sous-jacent à ces informations devrait permettre de mettre en évidence des excédents d'informations ou des manques d'informations au niveau des modèles, et ainsi de déterminer la nature du symptôme (vrai ou faux).

Il serait également pertinent d'intégrer les trois types de modification possibles sous le formalisme des réseaux de Petri au niveau du diagnostic : modification des coefficients de la matrice d'incidence C, modification de la dimension de la matrice d'incidence C et modification du marquage du réseau. Ceci pourrait permettre une modification du modèle du système indépendamment du type de symptôme à traiter ou du type d'erreur à corriger dans le cas des incohérences causées par les erreurs du modèle.

Or, l'étape de diagnostic devient une étape plus complexe quand ont prend en compte plusieurs types de modifications pour répondre à une incohérence. Cette étape devient alors critique, car plusieurs modifications peuvent être proposées et certaines modifications seront plus adaptées que autres, selon l'origine de l'incohérence. Il serait alors nécessaire de pouvoir exploiter les résultats entre la distinction des types de symptômes et aussi le résultat d'un étude postérieure dans le cas du type d'erreur de modélisation (excédent d'information et déficit d'information) pour proposer des modifications qui ne risquent pas de provoquer un blocage postérieur dans le modèle.

Par ailleurs, actuellement, dans l'approche proposée, lorsque la détection se fait via les deux modèles du système (modèle du comportement attendu et modèle de la commande) le diagnostic n'est pas capable de proposer des modifications pertinentes de modèles. Ceci vient du fait que le nombre de modifications envisageables est très important. Dans ces conditions, une solution serait de continuer l'observation du comportement du système sur un horizon de temps fixé et d'exploiter les observations ainsi obtenues pour restreindre l'ensemble des modifications utiles. Une autre solution serait d'envisager ou suspecter l'existence d'erreurs de modélisation dans le modèle de comportement réel MOD_{CR}, lequel est censé être correct. L'existence de ces erreurs pourrait être possiblement déterminé en intégrant à la méthode des mécanismes qui prennent en compte la connaissance d'expert du système ainsi que les spécifications de fonctionnement décrits dans le cahier de charges pour mettre en évidence ces erreurs.

Egalement la méthode de diagnostic peut proposer plusieurs modifications pour rétablir la cohérence. Or il serait insérant d'analyser chaque modification proposée par la méthode pour éliminer ces modifications qui au niveau du procédé ne semblent pas logiques même si au niveau du modèle sont possibles. Comme ça on pourrait assurer un ensemble de modifications qui restaurent la cohérence et qui en même temps proposent des modifications dans le procédé (si c'est le cas) qui correspondent bien à la réalité.

L'approche de détection et diagnostic que nous avons développée pourrait ainsi être un outil lors de la phase de conception et de mise en place des modèles. L'analyse des modèles telle que nous l'envisageons dans notre approche permet d'ajuster ces modèles en fonction du but de la modélisation. De cette façon, le modèle du comportement correct pourrait être complété avec des comportements défaillants bien identifiés et connus ou au contraire restreint aux comportements normaux en garantissant que les comportements décrits sont des comportements corrects. L'approche pourrait aussi être enrichie avec un

historique de modifications réalisées aux modèles afin d'identifier des erreurs communes réalisés dans la phase de conception des modèles, lesquels pourront alerter aux experts du système pour de nouvelles extensions des modèles mis en place.

Actuellement dans notre approche lors d'une incohérence dans un des modèles MOD_{CA} et MOD_C, le diagnostic réalise la modification de ce modèle pour le rendre à nouveau cohérent avec les observations. Néanmoins, pour l'instant rien ne garantit que les modifications réalisées sur les modèles ne modifient pas aussi la configuration partiellement identifiée jusque là. Une réflexion doit être menée pour analyser les implications des modifications des modèles sur la configuration.

A plus long terme, il serait intéressant de pouvoir intégrer notre approche dans une architecture de commande/surveillance distribuée dont nous avons montré les avantages au début de ce document. Des adaptations de la méthode sont nécessaires pour ce type d'architecture. Par exemple, chaque entité de diagnostic pourrait mettre en place cette approche de façon locale, mais il est alors nécessaire d'envisager des coopérations entre les différentes entités de diagnostic dans le but de maintenir la cohérence globale du modèle distribué. Ainsi cette coopération pourrait contribuer à la prise de décision locale basée sur les informations issues d'autres entités de diagnostic. Cette coopération permettrait, lors de la détection d'un symptôme par une entité, d'alerter d'autres entités chargées de parties du système susceptibles d'être touchées par une propagation de défaillance.

Les travaux que nous venons de présenter reposent sur l'hypothèse que les observations sont correctes. Ainsi, les observations ne sont jamais remises en cause et seuls les modèles du système sont susceptibles d'être incorrects. Il serait envisageable d'unifier nos travaux avec ceux menés dans le domaine du diagnostic embarqué pour l'automobile dans lesquels le rétablissement de la cohérence est envisagé au travers de la modification des séquences observées [Soldani *et al.*, 2006]. Ceci tendrait à définir un cadre générique du diagnostic des systèmes à événements discrets.

L'application de l'approche à un système de plus grande taille doit également être réalisée avec le but de mettre en évidence d'éventuelles adaptation de la méthode.

A terme, il faudra également que la partie diagnostic de notre approche soit généralisée de telle manière que cette étape puisse proposer des modifications générales applicables à plusieurs types de modèles (RdP, automates, etc.) utilisés pour la modélisation des systèmes à événements discrets.

Bibliographie

[Achour, 2005] Achour, Z. (2005). *Contribution à la synthèse des contrôleurs des systèmes à événements discrets partiellement observables.* Thèse de doctorat de l'Université Paul Verlaine Metz.

[Ashley et Holloway, 2001] Ashley, J. et Holloway, L. (2001). Diagnosis of conditions systems using diagnosis causal network. *IEEE International Conference on Systems, Man and Cybernetics*, pages 17–22.

[Ausfelder *et al.*, 1993] Ausfelder, C., Castelain, E. et Gentina, J. C. (1993). A hierarchical modular model of flexible manufacturing systems. *Proceedings on International Conference on Systems, Man and Cybernetics, IEEE*, 1:48–53.

[Babiceanu *et al.*, 2004] Babiceanu, R. F., Chen, F. F. et Sturges, R. H. (2004). Framework of the control of automates material handling systems using the holonic manufacturing approach. *International Journal of Productions Research ISSN 0020-7543*, 42(17):3551–3564.

[Baroni *et al.*, 2000] Baroni, P., Lamperti, G., Pogliano, G. et Zanella, M. (2000). Diagnosis of a class of distributed discrete event systems. *IEEE Transaction Systems, Man, and Cybernetics, Part A*, 30:736–752.

[Benielli et Cerato, 2001] Benielli, F. et Cerato, G. (2001). *Automatique informatique industrielle.* Editions Foucher ISBN-10-2216085901, Paris.

[Berruet *et al.*, 2002] Berruet, P., Craye, E. et Toguyeni, A. (2002). *Chapitre 6 de l'ouvrage Maîtrise de risques et sureté de fonctionnement des systèmes de production.* Hermes Science Publications, Paris.

[Besant, 1989] Besant, C. B. (1989). International journal of advanced manufacturing technology. *Springer-Verlag London*, 4(2).

[Betta et Pietrosant, 2000] Betta, G. et Pietrosant, A. (2000). Instrument fault detection and isolation, state of the art and new research trends. *IEEE Transaction on Instrumentation and measurements*, 49(1).

[Boubour et Jard, 1996] Boubour, R. et Jard, C. (1996). Une approche pour des capteurs d'alarmes intelligentes dans le réseaux. *Rapport de recherche ISNN 0249-6399.*

[Boufaied, 2004] Boufaied, A. (2004). *Contribution à la surveillance distribuée des systèmes à événements discrets complexes.* Thèse de doctorat de l'Université Paul Sabatier de Toulouse.

[Buchanan *et al.*, 1984] Buchanan, B. G., Shortliffe, E. H. et Wan, M. (1984). *Rule based expert systems, the MYCIN Experiments of the stanford heuristic programming project.* The Addison-Wesley Series in Artificial Intelligence ISBN 02-201101726.

[Cassandras, 1993] Cassandras, G. (1993). *Discrete events systems, modeling and performance analysis.* Aksen Associates Incorporated Publishers, ISBN-10-0256112126.

[Cassandras et Lafortune, 1999] Cassandras, G. et Lafortune, S. (1999). *Introduction to discrete events systems.* Kluwer academic Publishers ISBN-10-0792386094, Springer.

[Chaillet-Subias, 1995] Chaillet-Subias, A. (1995). Approche multi modèles pour la commande et la surveillance en temps réel des systèmes à événements discrets. *Thèse de doctorat de l'Université Paul Sabatier de Toulouse.*

[Chaillet-Subias *et al.*, 1997] Chaillet-Subias, A., Combacau, M., Courvoisier, M., de Bonneval, A., Sahraoui, A. E. K. et Zamaï, E. (1997). *Chapitre 3, section IV de l'ouvrage Concepts et outils pour les systèmes de production.* CEPADUES, Toulouse.

[Charbonnaud, 1992] Charbonnaud, P. (1992). Méthode d'aide au diagnostic curatif multimodèle. *Revue Européenne Diagnostic et sûreté de fonctionnement*, 2(1):61–90.

[Chung *et al.*, 2003] Chung, S., Wu, C. et Jeng, M. (2003). Failure diagnosis, a case study on modeling and analysis by petri nets. *IEEE International Conference on Systems, Man and Cybernetics*, 3:2727–2732.

[Combacau, 1991] Combacau, M. (1991). *Commande et surveillance des systèmes à événements discrets complexes, application aux ateliers flexibles.* Thèse de doctorat de l'Université Paul Sabatier de Toulouse.

[Combacau, 2002] Combacau, M. (2002). *Chapitre 11 de l'ouvrage Maîtrise de risques et sureté de fonctionnement des systèmes de production.* Hermes Science Publications, Paris.

[Combacau *et al.*, 2000] Combacau, M., Berruet, P., Zamai, E., Charbonnaud, E. et Khatab, A. (2000). *Supervision and monitoring of production systems.* Proceedings of the International Federation of Automatic Control, IFAC/IEEE MCPL, Grenoble, France.

[Darwiche, 1998] Darwiche, A. (1998). Model based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222.

[David et Halla, 1989] David, R. et Halla, H. (1989). *Du grafcet au réseaux de Petri.* Traité des Nouvelles Technologies série Automatique, Paris, Hermes.

[Davis, 1984] Davis, R. (1984). Diagnostic reasoning based on structure and behavior. *Exploring Artificial Intelligence, Morgan Kufmann Publishers Inc.*, 24(1-3):376–407.

[Davis et Hamscher, 1988] Davis, R. et Hamscher, W. (1988). Model based reasoning, troubleshooting. *Exploring Artificial Intelligence, Morgan Kufmann Publishers Inc.*, 32(1):297–346.

[Debouk *et al.*, 1998a] Debouk, R., Lafortune, S. et Teneketzis, D. (1998a). *A coordinated decentralized protocol for failure diagnosis of discrete event systems.* Proceeding of the Fourth Workshop on Discrete Event Systems , WODES, Londres, Inglaterre.

[Debouk *et al.*, 1998b] Debouk, R., Lafortune, S. et Teneketzis, D. (1998b). Coordinates decentralized protocols for failure diagnosis of discrete event systems. *Proceedings of the 37th IEEE Conference on Decision and Control*, 10-11:3763–3768.

[De Kleer, 2003] De Kleer, J. (2003). Fundamentals of model based diagnosis. *Proceedings of the International Federation of Automatic Control, IFAC-SafeProcess*, pages 25–36.

[De Kleer et Williams, 1987] De Kleer, J. et Williams, B. (1987). Diagnosing multiple faults. *Artificial Intelligence, Elsevier Science Publishers*, 32(2):97–130.

[De Kleer et Williams, 1989] De Kleer, J. et Williams, B. (1989). Diagnosis with behavioral modes. *Proceeding of the 11th International Joint Conference Artificial Intelligence*, pages 1324–1330.

[Diaz, 2001] Diaz, M. (2001). *Les réseaux de Petri, modèles fondamentaux.* Traité des Nouvelles Technologies série Automatique, Paris, Hermes.

[Dilts *et al.*, 1991] Dilts, D. M., Boyd, N. P. et Whorms, H. H. (1991). The evolution of control architectures for automated manufacturing systems. *Journal of Manufacturing systems*, 10(1):79–93.

[Dubuisson, 2001] Dubuisson, B. (2001). *Diagnostic, intelligence artificielle et reconnaissance des formes.* Science Publications. Paris, Hermes.

[Duffie, 1990] Duffie, N. A. (1990). *Synthesis of heterarchical manufacturing systems*, volume l4. Computer in Industry.

[Duffie *et al.*, 1988] Duffie, N. A., Chitturi, R. et Mou, J. I. (1988). Fault tolerant heterarchical control of heterogeneous manufacturing systems entities. *Journal of Manufacturing Systems*, 7(4):315–327.

[Duffie et Prabhu, 1996] Duffie, N. A. et Prabhu, V. V. (1996). Heterarchical control of higly distributed manufacturing systems. *International Journal Computer Integrated Manufacturing*, 9(4):270–281.

[Dvorak et Kuipers, 1989] Dvorak, D. et Kuipers, B. (1989). *Model based monitoring of dynamic systems.* Readings in Model-based diagnosis.

[FESTO,] FESTO. Fournisseur de maquettes à caractère didactique. URL : http ://www.festo.com.

[Gelgele et Wang, 1998] Gelgele, H. L. et Wang, K. (1998). An expert systems for engine fault diagnosis development and application. *Journal of Intelligent Manufacturing*, 9(6):539–545.

[Genc et Lafortune, 2003] Genc, S. et Lafortune, S. (2003). *Distributed diagnosis of discrete event systems using Petri nets*, volume 2679. In Application and Theory of Petri Nets, Series Lecture Notes in Computer Science.

[Genc et Lafortune, 2005] Genc, S. et Lafortune, S. (2005). A distributed algorithm for on-line diagnosis of place-bordered petri nets. *Proceeding of 16th International Federation of Automatic Control World Congress, IFAC.*

[Genesereth, 1984] Genesereth, M. R. (1984). *The use of design descriptions in automated diagnosis*, volume 24-No 1-3. Exploring Artificial Intelligence, Morgan Kufmann Publishers Inc.

[Gertler, 1998] Gertler, J. J. (1998). Survey of model-based failure detection and isolation in complex plants. *IEEE Control Systems Magazine*, 8(6):3–11.

[Ghazel, 2005] Ghazel, M. (2005). Surveillance des systèmes à événements discrets à l'aide des réseaux de petri t-temporels. *Thèse de doctorat de l'Université Lille.*

[Giard, 2003] Giard, V. (2003). *Gestion de la production et des flux.* Economica, Paris.

[Giua, 1997] Giua, A. (1997). Petri nets state estimators based on event observation. *Proceedings of the 36th Conference on Decision and Control (CDC)*, pages 4086–4091.

[Graton, 2005] Graton, G. (2005). Diagnostic des systèmes à l'aide d'observateurs à mémoire finie. application au common rail. *Thèse de doctorat de l'Université d'Orleans.*

[Harel, 1988] Harel, D. (1988). On visual formalisms. *Communications of the Association Computing Machinery*, 31(5):514–530.

[Hatvany, 1985] Hatvany, J. (1985). Intelligence and cooperation in heterarchic manufacturing systems. *Robotics Computer-Integrated Manufacturing*, 2(2):101–104.

[Henry, 2005] Henry, S. (2005). Synthèse de lois de commande pour la configuration et la reconfiguration des systèmes industriels complexes. *Thèse de doctorat de l'Institut National Polytechnique de Grenoble-INPG.*

[Hernandez De Leon, 2006] Hernandez De Leon, H. R. (2006). Supervision et diagnostic des procédés de production d'eau potable. *Thèse de doctorat de l'Institut National des Sciences Appliquées de Toulouse.*

[Ho, 1989] Ho, Y. C. (1989). Introduction to special issue on dynamics of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98.

[Holloway et Krogh, 1990] Holloway, L. et Krogh, B. (1990). Fault detection and diagnosis in manufacturing systems, a behavioral model approach. *IEEE Second International Conference on Computer Integrated Manufacturing*, pages 252–259.

[Huang, 1996] Huang, H. M. (1996). Intelligent manufacturing system control, reference model and initial implementation. *The 35th IEEE Conference on Decision and Control (CDC).*

[Isaza Narvaes *et al.*, 2006] Isaza Narvaes, C., Aguilar Martin, J., Le Lann, M. . V., Rios Bolivar, A. et Khatab, A. (2006). An optimization method for the data space partition obtained by classification techniques for the monitoring of dynamic processes. *9ème Congrès International de l'Association Catalane d'Intelligence Artificielle CCIA*, pages 80–87.

[Isermann, 1995] Isermann, R. (1995). Models based fault detection and diagnosis methods. *Proceedings on the American Control Conference*, pages 1605–1609.

[Jakobson et Weissman, 1993] Jakobson, G. et Weissman, M. (1993). Alarm correlation. *Network, IEEE*, 7(6):52–59.

[Jard *et al.*, 2005] Jard, C., Chatain, T. et Bourhis, P. (2005). Diagnostic temporel dans les systèmes répartis à l'aide de dépliage de réseaux de petri temporels. *Journal Européen des Systèmes Automatisés, JESA*, 39(1-2-3):351–365.

[Kamsu *et al.*, 2005] Kamsu, F. B., Chapurlat, V. et Prunet, F. (2005). Vérification de modèles de processus d'entreprise, une approche formelle. *Journal Européen des Systèmes Automatisés, JESA*, 39(9-10):1051–1078.

[Kelly et Bartlett, 2006] Kelly, E. M. et Bartlett, L. M. (2006). Application of digraph method in system fault diagnostic. *Proceeding of the First International Conference on Availability, Reliability and Security, IEEE Computer Society.*

[kempowsky, 2004] kempowsky, T. (2004). *Surveillance de procédés à base de méthodes de classification : conception d'un outil d'aide pour la détection et le diagnostic de défaillances.* Thèse de doctorat de l'Institut National des Sciences Appliquées, Toulouse, France.

[kim *et al.*, 2005] kim, S., Ahn, S. J., Chung, J., No, M. et Sin, S. (2005). *A rule based approach to network fault and security diagnosis with agent collaboration*, volume 3397. Lecture et Notes in Computer Science Springer Berling/ Heidelberg.

[Klein, 2005] Klein, S. (2005). Identification of discrete event systems for fault detection purpose. *Thèse de doctorat de l'Université de l'Ecole Normale Supérieure de Cachan.*

[Klein *et al.*, 2005] Klein, S., Lesage, J. J. et Litz, L. (2005). Identification comportementale des systèmes logiques en vue de leur surveillance. *Journal Européen des Systèmes Automatisés, JESA*, 39(1-2-3):111–126.

[Kramer et Senehi, 1993] Kramer, T. R. et Senehi, M. K. (1993). Feasibility study : reference architecture for machine control systems integration. *NISTIR 5517, Rapport du National Institute of Standards and Technology.*

[Kuzgunkaya et Elmaraghy, 2006] Kuzgunkaya, O. et Elmaraghy, H. (2006). Assessing the structural complexity of manufacturing systems configurations. *International Journal of Flexible Manufacturing Systems*, 18:145–171.

[LAAS-CNRS,] LAAS-CNRS. Time petri net analyser. URL : http ://www.laas.fr/tina/.

[Landou et Besançon-Voda, 2001] Landou, I. et Besançon-Voda, A. (2001). *Identification des systèmes.* Sciences Publications. Paris, Hermes.

[Lefebvre et Delherm, 2005] Lefebvre, D. et Delherm, C. (2005). Diagnosis with causality relationships and directed paths in pn models. *Proceeding of the 16th International Federation of Automatic Control IFAC World Congress.*

[Leitao et Restivo, 1999] Leitao, P. et Restivo, F. (1999). A layered approach to distributed manufacturing. *Proceedings of Interantional Conference, ASI.*

[Le Moigne, 1990] Le Moigne, J. (1990). *La modélisation des systèmes complexes.* Afcet Systèmes. Dunod, Patis Afcet Systèmes.

[Lopez Varela *et al.*, 2005] Lopez Varela, C., Subias, A. et Combacau, M. (2005). An adapted fmea based approach for failure diagnosis in distributed architectures. *World Congress Scientific Computation, Applied Mathematics and Simulation, IMACS.*

[Lopez Varela *et al.*, 2007] Lopez Varela, C., Subias, A. et Combacau, M. (2007). Approche de détection basée cohérence : modèles pour le diagnostic. *3ème Colloque International Francophone Performance et Nouvelles Technologies en Maintenance, PENTOM.*

[Magee et de Weck, 2004] Magee, C. L. et de Weck, O. L. (2004). Complexe system classification. *Fourteenth Annual International Symposium of the International Council On Systems Engineering (INCOSE).*

[McFarlane *et al.*, 1995] McFarlane, D., Marett, B., Elsley, G. et Jarvis, D. (1995). Application of holonic methodologies to problem diagnosis in a steel rod mill. *Proceedings of IEEE Interantional Conference on Systems, Man and Cybernetics*, 1:940–945.

[Meda *et al.*, 1998] Meda, M. E., Ramirez, A. et Malo, A. (1998). Identification in discrete event systems. *Actes de IEEE international Conference on Systems, Man and Cybernetics SMC*, 1:740–745.

[Meda *et al.*, 2005] Meda, M. E., Ramirez, A. et Malo, A. (2005). Identification of concurrent discrete event systems using petri nets. *Proceedings of the 17th World Congress Scientific Computation, Applied Mathematics and Simulation, IMACS*.

[Meida, 1997] Meida, D. M. (1997). *A model for alarm correlation intercommunication networks*. Thèse de doctorat de Federal University of Minas Gerais, Belo Horizonte, Brasil.

[Milne, 1987] Milne, R. (1987). Strategies for diagnosis. *IEEE transaction on systems, man and cybernetics*, 17(3).

[Muthukumar *et al.*, 1991] Muthukumar, C. T., Guarro, S. B. et Apostolakis, G. E. (1991). Logic flowgraph methodology : a tool for modeling embedded systems. *Proceedings of the IEEE 10th Digital Avionics Systems Conference*, pages 103–109.

[Niel et Craye, 2002] Niel, E. et Craye, E. (2002). *Maîtrise de risques et sureté de fonctionnement des systèmes de production*. Hermes Science Publications, ISBN-274620402, Paris.

[Olive, 2003] Olive, X. (2003). Approche intégrée à base de modèles pour le diagnostic hors ligne et la conception, application au domaine de l'automobile. *Thèse de doctorat de l'Université Paul Sabatier Toulouse*.

[Papadopoulos et McDermid, 2001] Papadopoulos, Y. et McDermid, J. (2001). Automated safety monitoring, a review and classification of methods. *International Journal of Condition Monitoring and Diagnosis Engineering Management*, 4(4):14–32.

[Pencolé, 2000] Pencolé, Y. (2000). Decentralized diagnoser approach, application to telecommunications networks. *Eleventh International Workshop on Principles of Diagnosis DX - 00*, pages 185–192.

[Peterson, 1981] Peterson, J. L. (1981). *Petri net theory and the modelling of systems*, volume 07632 de *Englewood cliffs*. Prentice Hall, ISBN-0136619835.

[Pinto-Leitão, 2004] Pinto-Leitão, P. J. (2004). An agile and adaptive holonic architecture for manufacturing control. *Thèse de doctorat de l'Institute Politechnique of Bargaça*.

[Poole, 1989] Poole, D. (1989). Normality and faults in logic-based diagnosis. *International Joint Conference on Artificial Intelligence, IJCAI*, pages 1304–1310.

[Ramadge et Wonham, 1989] Ramadge, P. J. G. et Wonham, W. M. (1989). The control of discrete event systems. *Proceeding of the IEEE*, 77(1):81–98.

[Reiter, 1987] Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–96.

[Samad *et al.*, 2007] Samad, T., Mclauglin, P. et Lu, J. (2007). Systems architecture for process automation. *Review and trends Journals of Process Control*, 17(3):191–201.

[Sampath *et al.*, 1995] Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K. et Tenketzis, D. C. (1995). Diagnosticability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575.

[Schnoeben, 1999] Schnoeben, P. (1999). *Techniques et outils du model-checking.* Viubert.

[Shiming *et al.*, 2000] Shiming, L., Gruver, W. A., Kotak, D. et Bardi, S. (2000). Holonic manufacturing systems for distributed control of automated guided vehicles. *IEEE International Conference on Systems, Man and Cybernetics*, 3:1727–1732.

[Soldani *et al.*, 2006] Soldani, S., Combacau, M., Thomas, J. et Subias, A. (2006). Intermittent fault detection trough message exchanges, a coherence based approach. *17th International Workshop on Principles of diagnosis DX - 06*, pages 251–256.

[Sourrisse et Boudillon, 1997] Sourrisse, C. et Boudillon, L. (1997). *La sécurité des machines automatisées. Tome 2 : Techniques et moyens de prévention opératifs -Systèmes de commande - Utilisation des machines.* Collection Technique, Groupe Schneider.

[Su *et al.*, 2002] Su, R., Wonham, W., Kurien, J. et Koutsoukos, S. (2002). Distributed diagnosis for qualitative systems. *Proceeding on the International Workshop on discrete event systems- WODES, IEEE Computer Society*, pages 169–174.

[Tabakow, 2007] Tabakow, I. (2007). *Using place invariants and test point to isolate faults in discrete event systems*, volume 13, No 2. Journal of Universal Computer Science, Springer.

[Tateno *et al.*, 2006] Tateno, S., Matsuyama, H. et Tsuge, Y. (2006). Fault diagnosis method using a signed digraph for multiple origins of failures-evaluations of the diagnosis accuracy. *Proceeding ot the IEEE International Conference on Control Applications*, pages 3271–3276.

[Tharumaraja *et al.*, 1998] Tharumaraja, A., Wells, A. J. et Nemes, L. (1998). Comparison of emerging manufacturing concepts. *Proceeding on the IEEE International Conference on Systems, Man and Cybernetics*, 1:325–331.

[Tharumarajah *et al.*, 1996] Tharumarajah, A., Welle, A. J. et Nemes, L. (1996). Comparison of the bionic, fractal and holonic manufacturing system concepts. *International Journal on Computer Integrated Manufacturing*, 9(3):217–226.

[Toguyeni, 1992] Toguyeni, A. (1992). Surveillance et diagnostic en ligne dans les systèmes flexibles de l'industrie manufacturière. *Thèse de doctorat de l'Université de Lille.*

[Toscano, 2005] Toscano, R. (2005). *Commande et diagnostic des systèmes dynamiques.* ellipses, Paris.

[Ushio *et al.*, 1998] Ushio, T., Onishi, I. et Okuda, K. (1998). Fault detection based on petri nets models with faulty behaviors. *IEEE Transactions on system, Man and Cybernetics*, 1:113–118.

[Valette et Künzle, 1994] Valette, R. et Künzle, L. A. (1994). Réseaux de petri pour la détection et le diagnostic. *Groupement de Recherche Automatique du CNRS, Journées d'Etude S3, Sûreté, Surveillance, Supervision, Détection et localisation de défaillance*, page 9.

[Villemeur, 1988] Villemeur, A. (1988). *Sûreté de fonctionnement des systèmes industriels*. Eyroles, Paris.

[Vu Trieu *et al.*, 2007] Vu Trieu, M., Nitin, A. et Wan, M. (2007). *Fault detection and control of process systems*, volume ID 80321. Hindaw Publishing Corporation Mahtematical Problems in Engineering.

[Y.Papadopoulos, 2002] Y.Papadopoulos (2002). *Model-based on-line monitoring using a state sensitive fault propagation model*, volume 2434 de *Englewood cliffs*. Springer Berlin Heidelberg.

[Zamaï, 1997] Zamaï, E. (1997). Architecture de surveillance-commande pour les systèmes à événements discrets complexes. *Thèse de doctorat de l'Université Paul Sabatier de Toulouse*.

[Zamaï *et al.*, 1998] Zamaï, E., Combacau, M. et Chaillet-Subias, A. (1998). Models and strategies for monitoring of flexible manufacturing systems. *Proceeding of the International Federation of Automatic Control, IFAC/INCOM*, pages 485–490.

[Zwingelstein, 1995] Zwingelstein, G. (1995). *Diagnostic de défaillances théorie et pratique pour les systèmes industriels*. Hermes Science Publications, ISBN-2866014634.

Annexe

Cette annexe contient des éléments de définition formelle des modèles utilisés dans l'exemple du chapitre 7. Ces notations parfois très lourdes (matrices des réseaux de Petri par exemple) ont été extraites du document principal pour ne pas le surcharger. Elles font toutefois l'objet d'une description exhaustive car elles sont le support de la méthode de modification des modèles proposée dans ces travaux.

Le modèle de comportement attendu MOD_{CA}

La structure du RdP modélisant le comportement attendu (noté RdP_{CA}) est décrite dans les matrices d'incidence du RdP.

La matrice d'incidence *Pre* est représentée dans la figure 20, *Post* est montrée dans la figure 21. Ces deux matrices composent la matrice d'incidence C de la figure 22.

Pre =

t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	p_1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	p_2
0	1	0	0	0	0	0	0	0	0	0	0	0	0	p_3
0	0	1	0	0	0	0	0	0	0	0	0	0	0	p_4
0	0	1	0	0	0	0	0	0	0	0	0	0	0	p_5
0	0	0	1	0	0	0	0	0	0	0	0	0	0	p_6
0	0	0	0	1	0	0	0	0	0	0	0	0	0	p_7
0	0	0	0	0	1	0	0	0	0	0	0	0	0	p_8
0	0	0	0	0	0	1	0	0	0	0	0	0	0	p_9
0	0	0	0	0	0	0	1	0	0	0	0	0	0	p_{10}
0	0	0	0	0	0	0	0	1	0	0	0	0	0	p_{11}
0	0	0	0	0	0	0	0	0	1	1	0	1	0	p_{12}
0	0	0	0	0	0	0	1	0	0	0	0	0	0	p_{13}
0	0	0	0	0	0	0	0	0	1	0	0	0	0	p_{14}
0	0	0	0	0	0	0	0	0	0	0	1	0	0	p_{15}
0	0	0	0	0	0	0	0	0	0	1	0	0	0	p_{16}
0	0	0	0	0	0	0	0	0	0	0	0	0	1	p_{17}
0	0	0	0	0	0	0	0	1	0	0	0	0	0	p_{18}

FIG. 20 – La matrice *Pre* du RdP_{CA}

Post =

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}
p_1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
p_2	0	1	0	0	0	0	0	0	0	0	0	0	0	0
p_3	1	0	0	0	0	0	0	0	0	0	0	0	0	0
p_4	0	1	0	0	0	0	0	0	0	0	0	0	0	0
p_5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
p_6	0	0	1	0	0	0	0	0	0	0	0	0	0	0
p_7	0	0	0	1	0	0	0	0	0	0	0	0	0	0
p_8	0	0	0	0	1	0	0	0	0	0	0	0	0	0
p_9	0	0	0	0	0	1	0	0	0	0	0	0	0	0
p_{10}	0	0	0	0	0	0	1	0	0	0	0	0	0	0
p_{11}	0	0	0	0	0	0	0	1	0	0	0	0	0	0
p_{12}	0	0	0	0	0	0	0	0	1	0	0	0	0	0
p_{13}	0	0	0	0	0	0	0	0	1	0	0	0	0	0
p_{14}	0	0	0	0	0	0	0	0	0	0	0	1	0	0
p_{15}	0	0	0	0	0	0	0	0	0	1	0	0	0	0
p_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	1
p_{17}	0	0	0	0	0	0	0	0	0	0	1	0	0	0
p_{18}	0	0	0	0	0	0	0	0	0	0	0	1	1	1

FIG. 21 – La matrice *Post* du RdP_{CA}

C =

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}
p_1	-1	0	0	0	0	0	0	0	0	0	0	1	1	1
p_2	-1	1	0	0	0	0	0	0	0	0	0	0	0	0
p_3	1	-1	0	0	0	0	0	0	0	0	0	0	0	0
p_4	0	1	-1	0	0	0	0	0	0	0	0	0	0	0
p_5	0	0	-1	0	0	0	0	1	0	0	0	0	0	0
p_6	0	0	1	-1	0	0	0	0	0	0	0	0	0	0
p_7	0	0	0	1	-1	0	0	0	0	0	0	0	0	0
p_8	0	0	0	0	1	-1	0	0	0	0	0	0	0	0
p_9	0	0	0	0	0	1	-1	0	0	0	0	0	0	0
p_{10}	0	0	0	0	0	0	1	-1	0	0	0	0	0	0
p_{11}	0	0	0	0	0	0	0	1	-1	0	0	0	0	0
p_{12}	0	0	0	0	0	0	0	0	1	-1	-1	0	-1	0
p_{13}	0	0	0	0	0	0	0	-1	1	0	0	0	0	0
p_{14}	0	0	0	0	0	0	0	0	0	-1	0	1	0	0
p_{15}	0	0	0	0	0	0	0	0	0	1	0	-1	0	0
p_{16}	0	0	0	0	0	0	0	0	0	0	-1	0	0	1
p_{17}	0	0	0	0	0	0	0	0	0	0	1	0	0	-1
p_{18}	0	0	0	0	0	0	0	0	-1	0	0	1	1	1

FIG. 22 – La matrice d'incidence C du RdP_{CA}

Le graphe de marquage associé au RdP_{CA} autre vue du modèle de comportement attendu MOD_{CA} est montré figure 23.

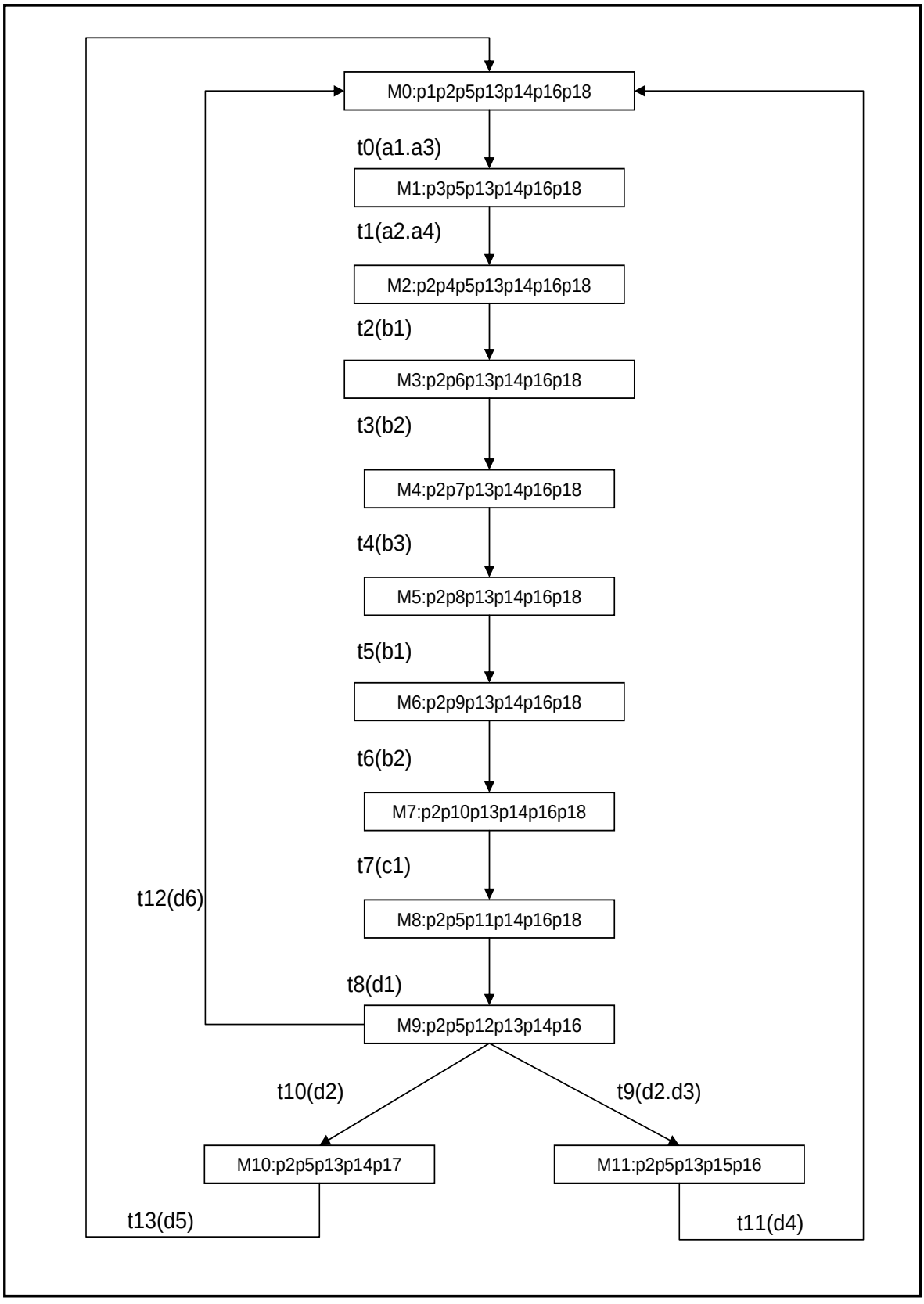


FIG. 23 – Modèle de comportement attendu

Le modèle de commande MOD_C

La structure du réseau de la commande RdP_C est donnée par les matrices d'incidence *Pre*, *Post* et C montrées dans les figures 24, 25 et 26, respectivement. Les événements associés à ces réseaux représentent les comptes-rendus attendus aux différentes commandes initiées par ce réseau.

Pre =

t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_2
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_3
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_4
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_5
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_6
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_7
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_8
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	p_9
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	p_{10}
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	p_{11}
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	p_{12}
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	p_{13}
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	p_{14}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	p_{15}
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	p_{16}
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	p_{17}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	p_{18}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	p_{19}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	p_{20}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	p_{21}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	p_{22}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	p_{23}
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_{24}
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	p_{25}
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	p_{26}

FIG. 24 – La matrice *Pre* du RdP_C

Le modèle de la commande MOD_C contient l'ensemble des marquages accessibles suivant M_C =$\{M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8, M_9, M_{10}, M_{11}, M_{12}, M_{13}, M_{14}, M_{15}, M_{16}, M_{17}, M_{18}\}$. La figure 27 montre MOD_C.

Post =

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	p_1
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_2
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_3
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_4
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_5
	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_6
	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_7
	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_8
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_9
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	p_{10}
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	p_{11}
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	p_{12}
	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	p_{13}
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	p_{14}
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	p_{15}
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	p_{16}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	p_{17}
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	p_{18}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	p_{19}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	p_{20}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	p_{21}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	p_{22}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	p_{23}
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	p_{24}
	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	p_{25}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	p_{26}

FIG. 25 – La matrice *Post* du RdP_C

C =

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	
	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	p_1
	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_2
	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_3
	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_4
	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_5
	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_6
	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_7
	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_8
	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	p_9
	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	p_{10}
	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	p_{11}
	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	p_{12}
	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	0	0	p_{13}
	0	0	0	0	0	0	0	0	0	0	0	1	-1	-1	0	0	-1	0	0	0	0	p_{14}
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	-1	0	0	0	0	0	0	p_{15}
	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0	0	p_{16}
	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0	p_{17}
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	-1	0	0	0	0	0	p_{18}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	p_{19}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0	p_{20}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	p_{21}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	p_{22}
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1	p_{23}
	0	0	0	-1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	p_{24}
	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	p_{25}
	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	1	1	p_{26}

FIG. 26 – La matrice d'incidence C du RdP_C

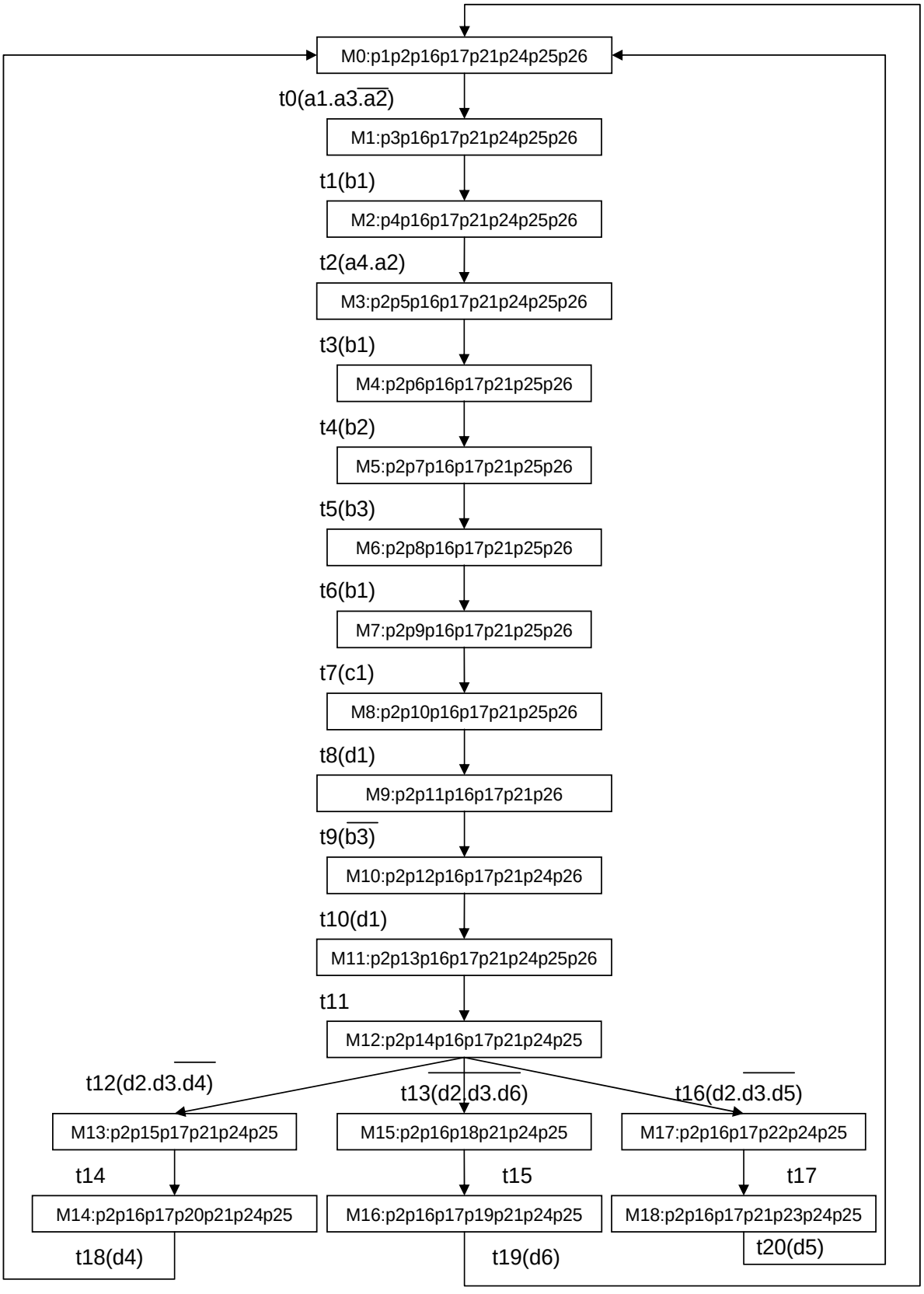


FIG. 27 – Modèle de la commande

Les différentes configurations des trois ensembles

Nous présentons dans la figure 28 les 109 configurations pour les trois modèles.

N°	Sous-ensembles	N°	Sous-ensembles	N°	Sous-ensembles
1.	<ACR>	38	<R>, <AC>, <ARC>,	75	<R>, <AR>, <RC>, <ARC>,
2	<A>, <RC>,	39	<R>, <AR>, <RC>,	76	<C>, <AC>, <AR>, <RC>,
3	<A>, <ARC>,	40	<R>, <AR>, <ARC>,	77	<C>, <AC>, <AR>, <ARC>,
4	<R>, <AC>,	41	<R>, <RC>, <ARC>	78	<C>, <AC>, <RC>, <ARC>,
5	<R>, <ARC>,	42	<AC>, <AR>, <RC>,	79	<C>, <AR>, <RC>, <ARC>,
6	<C>, <AR>	43	<AC>, <AR>, <ARC>,	80	<AC>, <AR>, <RC>, <ARC>,
7	<C>, <ARC>,	44	<AC>, <RC>, <ARC>	81	<A>, <AC>, <AR>, <RC>, <ARC>
8	<AR>, <AC>,	45	<AR>, <RC>, <ARC>	82	<R>, <C>, <AC>, <AR>, <RC>
9	<AR>, <RC>,	46	<A>, <R>, <C>, <AC>,	83	<R>, <C>, <AC>, <AR>, <ARC>
10	<AR>, <ARC>,	47	<A>, <R>, <C>, <AR>,	84	<R>, <C>, <AC>, <RC>, <ARC>
11	<AC>, <RC>,	48	<A>, <R>, <C>, <RC>,	85	<R>, <C>, <AR>, <RC>, <ARC>
12	<AC>, <ARC>,	49	<A>, <R>, <C>, <ARC>,	86	<R>, <AC>, <AR>, <RC>, <ARC>
13	<RC>, <ARC>,	50	<A>, <R>, <AC>, <AR>,	87	<C>, <AC>, <AR>, <RC>, <ARC>
14	<A>, <C>, <R>,	51	<A>, <R>, <AC>, <RC>,	88	<A>, <R>, <C>, <AC>, <AR>
15	<A>, <C>, <AR>,	52	<A>, <R>, <AC>, ARC>,	89	<A>, <R>, <C>, <AC>, <RC>
16	<A>, <C>, <RC>,	53	<A>, <R>, <AR>, <RC>,	90	<A>, <R>, <C>, <AC>, <ARC>
17	<A>, <C>, <ARC>,	54	<A>, <R>, <AR>, <ARC>,	91	<A>, <R>, <C>, <AR>, <RC>
18	<A>, <R>, <AC>,	55	<A>, <R>, <RC>, <ARC>,	92	<A>, <R>, <C>, <AR>, <ARC>
19	<A>, <R>, <RC>,	56	<A>, <C>, <AC>, <AR>,	93	<A>, <R>, <C>, <RC>, <ARC>
20	<A>, <R>, <ARC>,	57	<A>, <C>, <AC>, <RC>,	94	<A>, <R>, <AC>, <AR>, <RC>
21	<A>, <AC>, <AR>,	58	<A>, <C>, <AC>, <ARC>,	95	<A>, <R>, <AC>, <AR>, <ARC>
22	<A>, <AC>, <RC>,	59	<A>, <C>, <AR>, <RC>,	96	<A>, <R>, <AC>, <RC>, <ARC>
23	<A>, <AC>, <ARC>,	60	<A>, <C>, <AR>, <ARC>,	97	<A>, <R>, <AR>, <RC>, <ARC>
24	<A>, <AR>, <RC>,	61	<A>, <C>, <RC>, <ARC>,	98	<A>, <C>, <AC>, <AR>, <RC>
25	<A>, <AR>, <ARC>,	62	<A>, <AC>, <AR>, <RC>,	99	<A>, <C>, <AC>, <AR>, <ARC>
26	<A>, <RC>, <ARC>,	63	<A>, <AC>, <AR>, <ARC>,	100	<A>, <C>, <AC>, <RC>, <ARC>
27	<C>, <R>, <AC>,	64	<A>, <AC>, <RC>, <ARC>,	101	<A>, <R>, <C>, <AC>, <AR>, <RC>,
28	<C>, <R>, <AR>,	65	<A>, <AR>, <RC>, <ARC>,	102	<A>, <R>, <C>, <AC>, <AR>, <ARC>,
29	<C>, <R>, <ARC>,	66	<R>, <C>, <AC>, <AR>,	103	<A>, <R>, <C>, <AC>, <AR>, <ARC>,
30	<C>, <AC>, <AR>,	67	<R>, <C>, <AC>, <RC>,	104	<A>, <R>, <C>, <AC>, <RC>, <ARC>,
31	<C>, <AC>, <RC>,	68	<R>, <C>, <AC>, <ARC>,	105	<A>, <R>, <C>, <AR>, <RC>, <ARC>,
32	<C>, <AC>, <ARC>,	69	<R>, <C>, <AR>, <RC>,	106	<A>, <R>, <AC>, <AR>, <RC>, <ARC>,
33	<C>, <AR>, <RC>,	70	<R>, <C>, <AR>, <ARC>,	107	<A>, <C>, <AC>, <AR>, <RC>, <ARC>,
34	<C>, <AR>, <ARC>,	71	<R>, <C>, <RC>, <ARC>,	108	<R>, <C>, <AC>, <AR>, <RC>, <ARC>,
35	<C>, <RC>, <ARC>,	72	<R>, <AC>, <AR>, <RC>,	109	<A>, <R>, <C>, <AC>, <AR>, <RC>, <ARC>,
36	<R>, <AC>, <AR>,	73	<R>, <AC>, <AR>, <ARC>,		
37	<R>, <AC>, <RC>,	74	<R>, <AC>, <RC>, <ARC>,		

FIG. 28 – Liste de configurations

Auteur : Carmen Guadalupe LOPEZ-VARELA

Titre : Détection et diagnostic basés cohérence pour les systèmes à événements discrets : vers la prise en compte des erreurs de modélisation

Directeurs de thèse : Mme. Audine SUBIAS et M. Michel COMBACAU

Lieu et date de soutenance : LAAS-CNRS, 17 décembre 2007

Résumé : Cette thèse propose une méthode de détection et diagnostic basée cohérence pour les systèmes à événements discrets. La méthode prend en compte la possibilité d'erreurs dans les modèles utilisés comme référence du bon comportement du système. La détection est réalisée par la vérification de cohérence entre le comportement observé et le comportement décrit par les modèles du système. Dans la mesure où les modèles utilisés ne sont pas exempts d'erreurs, la perte de cohérence détectée peut correspondre à une réelle défaillance au niveau du procédé ou à une erreur dans les modèles. Pour distinguer ces deux cas, il est nécessaire de déterminer les sous-modèles communs. Chaque observation émanant du procédé apporte une information utile pour l'identification de la ou des types de configuration entre les modèles. L'identification est réalisée en ligne et repose sur le parcours d'un graphe décrivant exhaustivement l'ensemble des configurations des modèles. Enfin, le rétablissement de la cohérence avec les observations par modification des modèles constitue le principe du diagnostic.

Mots clés : Systèmes à événements discrets, détection, diagnostic, vérification de cohérence, modèles comportementaux, erreurs de modélisation, réseau de Petri, identification des configurations des modèles.

Discipline : Systèmes Industriels

Consistency-based detection and diagnosis for discrete event systems : towards the consideration of modelling errors

Abstract : This thesis proposes a consistency-based detection and diagnosis method for discrete event systems. The method takes into account the possibility of errors within the models that are used as a reference of the correct behaviour of the system. The detection step is carried out by consistency verification between the observed behaviour and the one described in the system models. Since the models may be incorrect, detected inconsistencies can either correspond to failures of the process or to modelling errors. To distinguish between these two cases, it is necessary to determine the joint sub-models. Each observation brings useful information for the configuration identification between models. The on-line identification is based on a search in a graph that exhaustively describes the possible configurations of the models. Finally, the diagnosis principle consists in restoring the consistency with the observations by modifying the models.

Keywords : Discrete event systems, detection, diagnosis, consistency, behavioural models, modelling errors, Petri nets, configuration identification