CONTRIBUTION AU DIAGNOSTIC DECENTRALISE DES SYSTEMES A EVENEMENTS DISCRETS : APPLICATION AUX SYSTEMES MANUFACTURIERS

Philippot Alexandre

UNIVERSITE DE REIMS CHAMPAGNE ARDENNE

THESE

Présentée pour obtenir le grade de

Docteur de l'Université de Reims Champagne Ardenne

Spécialité

GENIE INFORMATIQUE, AUTOMATIQUE ET TRAITEMENT DU SIGNAL

Par

Alexandre PHILIPPOT

CONTRIBUTION AU DIAGNOSTIC DECENTRALISE DES SYSTEMES A EVENEMENTS DISCRETS : APPLICATION AUX SYSTEMES MANUFACTURIERS.

Soutenue publiquement le 18 juillet 2006, devant le jury composé de :

Rapporteurs	M. Jean-Jacques LESAGE	Professeur, LURPA – ENS de Cachan
	M. Etienne CRAYE	Professeur, Ecole Centrale de Lille
Examinateurs	M. Michel COMBACAU	Professeur, Université de Toulouse III
	M. Bernard RIERA	Professeur, Université de Reims
	Mme Véronique CARRE-MENETRIER	Professeur, Université de Reims (Directrice de thèse)
	M. Moamar SAYED-MOUCHAWEH	Maître de Conférences, Université de Reims (Co-directeur de thèse)

REMERCIEMENTS

Les travaux présentés dans ce mémoire ont été réalisés au sein de l'équipe Systèmes à Evénements Discrets – Supervision du CReSTIC (Centre de Recherche en STIC) de l'Université de Reims Champagne Ardenne. Je remercie donc Monsieur le Professeur Janan ZAYTOON, directeur du CReSTIC, de m'avoir accueilli au sein du laboratoire, mais également pour m'avoir encouragé dans mes recherches dès le DEA et soutenu tout au long de ces années.

Ce mémoire est également pour moi l'occasion de remercier deux personnes qui m'ont toujours épaulées et m'ont permises d'arriver jusqu'ici. Tout d'abord, je tiens à exprimer ma reconnaissance profonde à Madame le Professeur Véronique CARRE-MENETRIER qui a dirigée mes travaux, m'a accordée sa confiance et m'a soutenue sur de nombreux plans durant cette thèse. Je la remercie grandement que ce soit pour ces qualités scientifiques ou humaines, faisant toujours preuve de réflexion, de patience et de diplomatie. Je souhaiterai également indiquer l'importance de Monsieur Moamar SAYED-MOUCHAWEH, Maître de conférences et co-encadrant de cette thèse, pour son engouement, sa disponibilité, ses conseils et son amitié.

Je remercie vivement les rapporteurs de ce mémoire de thèse, Monsieur le Professeur Jean-Jacques LESAGE de l'Ecole Normale Supérieure de Cachan, Directeur du Laboratoire Universitaire de Recherche en Production Automatisée (LURPA), et Monsieur le Professeur Etienne CRAYE, Directeur de l'Ecole Centrale de Lille et chercheur au Laboratoire d'Automatique, Génie Informatique & Signal (LAGIS). Je leur suis très reconnaissant pour avoir bien voulu consacrer de leur temps à l'étude de mes travaux. Merci pour leurs remarques et suggestions durant nos différentes rencontres.

Un grand merci à Monsieur Michel COMBACAU, Professeur à l'Université de Toulouse III et chercheur au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS – CNRS), de m'avoir fait l'honneur d'examiner mes travaux et d'avoir accepté de présider ce jury.

J'ai été très heureux de compter parmi mon jury Monsieur Bernard RIERA, Professeur de l'Université de Reims Champagne Ardenne. Je le remercie de son regard extérieur, pertinent et critique sur mon travail.

Bien évidemment, je tiens également à remercier les membres du laboratoire mais surtout l'ensemble de l'équipe SED – Supervision (Abdelouahed, Benoît, Pascale, Bernard, François, Nadhir, Moamar et Véro) et tous ceux que j'oublie.

Je tiens à exprimer ma reconnaissance aux membres du département EEA de la Faculté des Sciences Exactes et Naturelles et au département GCE de l'IUT de Reims pour m'avoir confié des enseignements pendant ces années.

Un merci particulier aux joueurs (Fabien, François, Damien, …) et aux pseudo-joueurs de cartes (Cyril, Alex, Benoît, …) pour les leçons données et reçues à la belote et au tarot.

Evidemment, je ne peux oublier les LAMpins (Beevis, Doc, Jeff et Val) pour « La vaisselle » et autres expériences insensées et non avouables. Pour tous les moments de bonne

humeur et de fous rires passés ensemble.

Et puis je tiens à finir ces remerciements par ceux qui m’ont accompagnés tous les jours depuis le début et à qui je souhaite dédier ce mémoire de thèse. Je pense évidemment à mes parents, mon frère et ma belle-sœur, ma belle famille et puis à Aurélie. Ma petite femme adorée qui m’a supportée dans les deux sens du terme. Sa présence, son regard et le miracle qu’elle porte en elle fait d’elle un être d’exception.

Il est coutume sur cette page de dédier le mémoire à quelqu'un ou de mettre une citation célèbre. Ayant déjà fait une dédicace, j'ai souhaité ici faire référence à un humoriste disparu qui, très **discrètement**, est devenu un **événement** incontournable de notre société.

"L'œil du sourd est normal" (Pierre Desproges)

TABLE DES MATIERES

INTRODUCTION GENERALE

Ce mémoire apporte une contribution au diagnostic décentralisé des Systèmes à Evénements Discrets (SED) essentiellement pour les applications manufacturières.

Le diagnostic des défauts (Fault Detection and Isolation) des systèmes industriels est à l'origine de nombreux travaux depuis les dernières années. Il est défini comme l'opération permettant de détecter un défaut, de localiser son origine et de déterminer ses causes. Son principe général consiste à confronter les données relevées au cours du fonctionnement réel du système avec la connaissance dont on dispose sur son fonctionnement normal ou anormal. L'intérêt pour le diagnostic des défaillances s'explique par la complexité croissante des systèmes industriels qui sont de plus en plus exigeants en terme de contraintes de sécurité, de fiabilité, de disponibilité et de performances. En fait, la possibilité qu'un système tombe en panne croît malgré les précautions de manipulations, le développement de techniques de conception de la commande et l'expérience des Opérateurs Humains de Supervision (OHS) [SAM 94].

Par conséquent, un module de diagnostic est nécessaire pour empêcher la propagation de pannes et pour limiter leurs conséquences qui peuvent être catastrophiques au niveau économique mais aussi au niveau environnemental [PER 84]. Un autre avantage des modules de diagnostic est de faciliter aux Opérateurs Humains de Supervision, la tâche de supervision en ligne des systèmes complexes [TRI 04]. En effet, ces OHS ont une réelle difficulté à lier un défaut à son origine et à déterminer ses causes en raison du nombre élevé de variables corrélées de manière non apparente et complexe.

Il existe un grand nombre de méthodes de diagnostic [BOU 96], [COM 91], [DEB 00], [DUB 01], [FRA 90], [GAR 05], [GEN 03], [HOL 94], [LIN 94], [PAN 00], [PHI 05a], [SAM 01], [MOU 02], [SU 04], [WAN 00]... Elles se basent sur un modèle du comportement normal et/ou défaillant du système et se distinguent selon différents critères : la dynamique du procédé à diagnostiquer (discret, continu ou hybride), l'implémentation du diagnostic en ligne et/ou hors ligne, la complexité du procédé, la nature de l'information disponible (qualitative et/ou quantitative), sa profondeur (analytique ou heuristique) et sa distribution (centralisée ou décentralisée). En général, ces méthodes peuvent être divisées en deux catégories :

- méthodes externes : qui sont des méthodes à base de connaissances, des méthodes empiriques et de traitement du signal,
- méthodes internes qui représentent des méthodes à base de modèles quantitatifs et/ou qualitatifs.

Nous avons restreint le domaine d'étude aux méthodes de diagnostic à base de modèles qualitatifs représentés comme des Systèmes à Evénements Discrets. Cependant, pour réaliser une approche de diagnostic des SED, il faut lever un certain nombre de difficultés rencontrées au niveau de :

La modélisation du procédé

Que ce soit du côté Partie Opérative (PO), modélisant l'ensemble des réactions possibles du procédé en fonction de sa technologie, ou du côté Partie Commande (PC) pour la modélisation du comportement désiré, il faut pouvoir choisir l'outil de modélisation pertinent et le niveau de modélisation ou d'abstraction adéquat (GRAFCET, Réseaux de Petri, automates à états, …).

La structure de la prise de décision

Il existe plusieurs structures de prise de décision dépendant de la distribution de l'information disponible sur le procédé : centralisée, décentralisée ou distribuée. La structure centralisée fournit les meilleures performances en terme de diagnostic mais elle s'expose au problème d'explosion combinatoire. La structure décentralisée diminue ce risque d'explosion mais elle ne permet pas toujours de diagnostiquer l'ensemble des défauts diagnosticables par un diagnostiqueur. C'est pour cette raison qu'un coordinateur est très souvent nécessaire. Il permet de fusionner les décisions locales de tous les diagnostiqueurs, et définit un ensemble de règles nécessaires pour le diagnostic des défauts non diagnosticables par aucun des diagnostiqueurs locaux. La structure distribuée diminue également l'explosion combinatoire et évite l'utilisation d'un coordinateur grâce à l'utilisation d'un protocole de communication entre les différents diagnostiqueurs locaux. Le protocole est souvent complexe et s'expose à un problème de retard de communication.

La modélisation des défauts

Plusieurs méthodes de modélisation des défauts existent en fonction des modèles qui sont à diagnostiquer : les modèles à base d'événements, les modèles à base d'états ou les modèles dits mixtes, c'est-à-dire mélangeant les deux notions précédentes. Les modèles à base d'événements considèrent le défaut comme l'exécution d'un événement. La détection et l'isolation sont réalisées en utilisant uniquement les séquences d'événements observables. Ces modèles ont l'avantage de détecter les défauts intermittents mais ils nécessitent l'initialisation du diagnostiqueur en même temps que le procédé, ce qui est difficile à obtenir dans le cas des procédés manufacturiers. Les modèles à base d'états modélisent un défaut par l'atteignabilité à un état défaillant. Ils détectent et isolent le défaut en observant les états caractérisés par les sorties statiques des capteurs et les commandes envoyées par la Partie Commande (PC). Par contre, ils ne peuvent pas diagnostiquer les défauts intermittents mais ont l'avantage de ne pas avoir besoin de la synchronisation à l'initialisation. Les modèles mixtes utilisent une modélisation à base d'événements et à base d'états afin de cumuler les avantages de chacun des modèles.

La notion de diagnosticabilité

Cette notion permet de vérifier si chaque défaut, appartenant à l'ensemble prédéfini des défauts, peut être détecté et isolé dans un temps fini après son occurrence. Elle est définie en utilisant soit un modèle du procédé à base d'événements soit un modèle à base d'états, et cela en fonction de la structure de la prise de décision choisie.

Contribution de la thèse

Les travaux de ce mémoire portent sur la proposition d'une approche de diagnostic des Systèmes à Evénements Discrets (SED) pour les systèmes manufacturiers par la cohabitation de toutes les informations disponibles sur un procédé [MOU 05], [PHI 05a], [PHI 05d].

Le GRAFCET a été choisi afin de représenter l'information de la Partie Commande car il constitue un outil très utilisé pour les systèmes manufacturiers. En ce qui concerne

l'information fournie par la Partie Opérative (PO), nous avons opté pour les automates à états permettant une représentation bas niveau du comportement du procédé et l'utilisation d'outils de composition synchrone et de projection naturelle. Afin de renforcer l'information nécessaire au diagnostic des SED, nous avons également intégré une information temporelle sur la réactivité des actionneurs à travers des fonctions de prévision basées sur la logique floue permettant d'informer l'utilisateur sur l'usure des éléments de la PO.

La nature de l'information présente sur les systèmes manufacturiers étant décentralisée, nous avons choisi une structure décentralisée avec coordinateur. Afin de pouvoir diagnostiquer l'ensemble des défauts susceptibles de survenir sur un procédé, nous avons choisi une modélisation mixte, à base d'événements et à base d'états mais faisant appel également à l'information temporelle correspondant à la dynamique du procédé. Les diagnostiqueurs locaux détectent et isolent un défaut soit par l'observation d'une évolution du procédé qui n'aurait pas dû se produire dans un contexte donné, soit par l'observation de la violation des contraintes temporelles qui se traduisent par l'absence ou l'occurrence d'un événement en dehors d'un intervalle de temps correspondant à la dynamique du procédé.

Enfin, et pour vérifier que la structure décentralisée avec coordinateur procède au diagnostic de l'ensemble des défauts diagnosticables, nous avons mis en œuvre une notion de co-diagnosticabilité mixte (à base d'événements, à base d'états et à base d'informations temporelles). Cette notion permet de garantir que chaque défaut appartenant à l'ensemble des défauts est diagnosticable par au moins un diagnostiqueur local ou par le coordinateur dans un délai fini. La Figure 1 illustre les six étapes de construction hors ligne de l'approche de diagnostic.

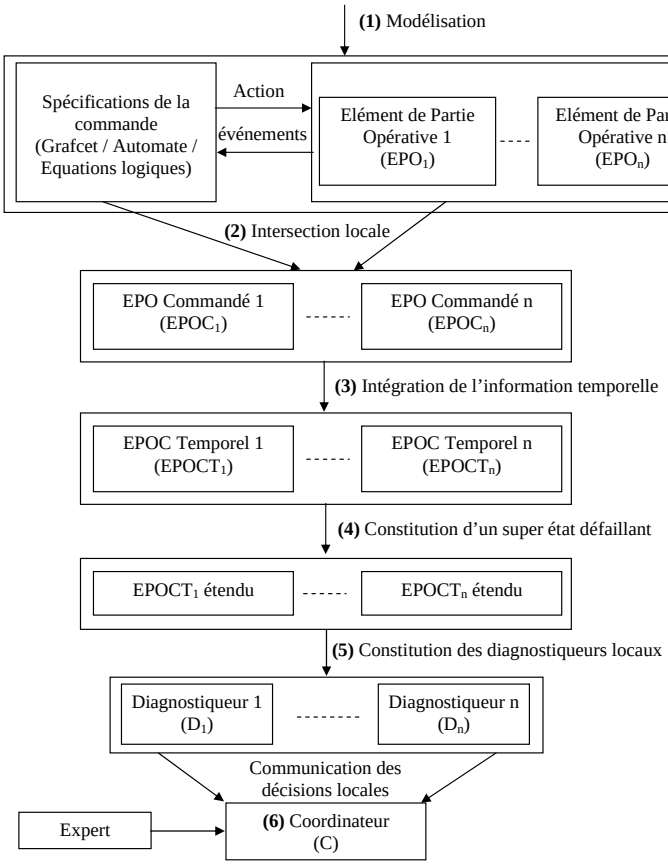


Figure 1 : Constitution hors ligne de l'approche proposée de diagnostic

1) La première étape consiste à modéliser la PC et la PO. La commande est modélisée en GRAFCET et considérée comme "optimale" en terme de sûreté de fonctionnement [NDJ 99], [TAJ 05]. La modélisation de la PO est modulaire. Chaque élément du procédé est modélisé en Elément de Partie Opérative EPO_i, $i \in \{1, 2, \ldots, n\}$ où *n* est le nombre d'éléments formant le procédé.

2) La deuxième étape consiste à extraire le comportement désiré localement sur chaque EPO_i. Pour cela, un algorithme d'intersection locale est appliqué entre l'EPO_i et la commande. Cette intersection conduit à l'obtention des Eléments de Partie Opérative Commandés : $EPOC_i$, $i \in \{1, 2, \ldots, n\}$.

3) L'intégration de l'information temporelle à chaque $EPOC_i$ est réalisée en étape 3 sous forme de fonctions de prévision introduites à chaque état. Cette intégration conduit à l'obtention des EPOC Temporels (EPOCT).

4) L'ajout d'un super état de défaut, qui peut être atteint suite à l'occurrence d'un défaut, à chaque $EPOCT_i$ constitue l'étape 4. Il en résulte un $EPOCT_i$ étendu avec $i \in \{1, 2, \ldots, n\}$.

5) L'étape 5 est la constitution des diagnostiqueurs locaux à partir des EPOCT étendus en ajoutant des étiquettes à chaque état de chaque $EPOCT_i$ étendu. Ces étiquettes indiquent si un défaut a eu lieu et renseignent sur sa localisation.

6) La dernière étape consiste à réaliser un coordinateur devant garantir un diagnostic équivalent à celui du diagnostiqueur global. Ce coordinateur utilise un ensemble de règles de fusion des décisions locales envoyées par les diagnostiqueurs locaux. Le but de ces règles est de lever les ambiguïtés et les cas d'indécision liés à l'observation partielle du procédé par chacun des diagnostiqueurs locaux.

La construction de l'approche de diagnostic décentralisé avec coordinateur s'effectue hors ligne. L'implantation en ligne de cette approche s'effectue par l'intégration des diagnostiqueurs et du coordinateur dans un filtre intercalé entre une commande implantée en ligne par l'utilisateur et la PO (Figure 2). Cette commande est réalisée par l'utilisateur ce qui implique une possibilité de défauts provenant d'une erreur de la commande en plus des défauts issus de la PO (actionneurs et capteurs).

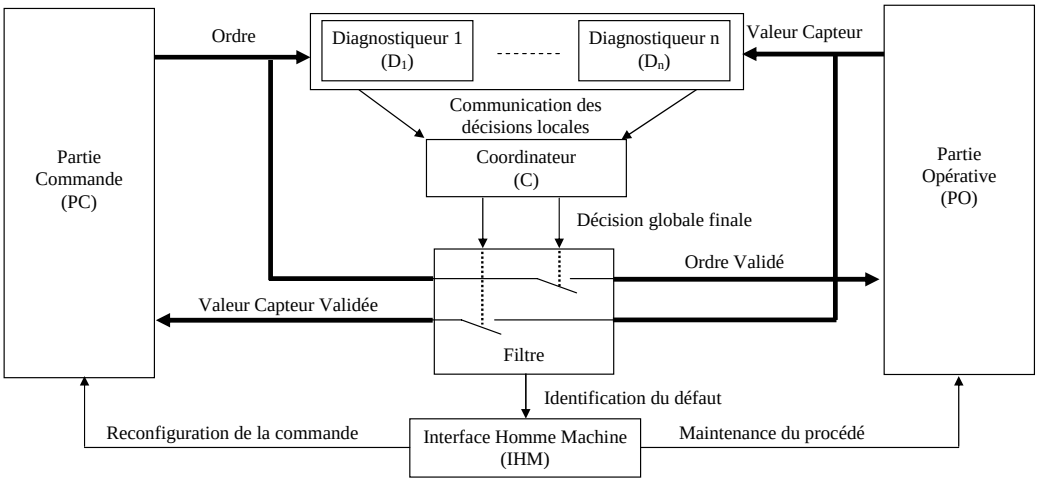


Figure 2 : Implantation en ligne de l'approche de diagnostic proposée dans la thèse

Organisation du mémoire de thèse

Le premier chapitre est un chapitre introductif qui vise à rappeler, dans un premier temps, la terminologie du diagnostic dans la littérature et celle adoptée dans ce mémoire. Ensuite, nous proposons une classification des principales méthodes de diagnostic abordées dans la littérature.

Le chapitre 2 présente un état de l'art des méthodes de diagnostic des défauts pour les SED, l'annexe 1 en fournit les caractéristiques principales, les outils de modélisation et les notations correspondantes. Nous présentons dans ce chapitre, les différentes méthodes de diagnostic de SED évoquées dans la littérature et nous introduisons une comparaison selon plusieurs critères dans le but de justifier les choix retenus dans la thèse pour : l'outil de modélisation (les automates), la structure de la prise de décision (décentralisée) et le type de modélisation des défauts (à base d'événements, à base d'états et à base d'informations temporelles). Nous montrons que le diagnostic nécessite l'exploitation de toute l'information disponible afin de réaliser un modèle riche du procédé à diagnostiquer.

Le troisième chapitre présente la démarche globale de diagnostic décentralisé des SED. Les différentes étapes, nécessaires pour la construction hors ligne des diagnostiqueurs locaux ainsi que pour le coordinateur, sont détaillées et expliquées autour d'un exemple. L'étape de modélisation de la PO est notamment développée afin de montrer les difficultés de cette modélisation. Une méthodologie de modélisation modulaire de la PO a permis d'aboutir à une bibliothèque des différents éléments présents sur un système manufacturier (Annexe 2).

Le dernier chapitre est consacré à l'application de la démarche sur deux exemples de systèmes manufacturiers. Nous montrons, sur le premier exemple, l'efficacité et l'intérêt de la structure décentralisée par rapport à une structure centralisée en terme d'explosion combinatoire. Ensuite, nous présentons une application complexe utilisant différentes technologies d'éléments de PO. Enfin, nous terminons ce chapitre par le test de cette approche en utilisant un outil de simulation à base de *Stateflow* de MATLAB.

Nous terminons ce mémoire de thèse par les conclusions sur nos travaux et les perspectives de recherche associées.

Chapitre 1 : Diagnostic des défaillances : Définitions et notations

1.1. Introduction

Ce chapitre est un chapitre introductif visant à rappeler, dans un premier temps, la terminologie utilisée pour le diagnostic des défaillances, rencontrée dans la littérature et retenue dans ce mémoire. Il met en place les concepts de surveillance, de diagnostic et de supervision et situe la place de la détection pour l'ensemble du mémoire de thèse. Ensuite, nous proposons une classification des principales méthodes de diagnostic retrouvées dans la littérature. Cette classification est réalisée selon la nature et la profondeur de l'information disponible, de la dynamique du procédé et de sa structure de décision. Ce chapitre permet de voir globalement les bases du diagnostic des défaillances du vocabulaire jusqu'à l'implantation. L'objectif est de positionner, parmi les méthodes existantes de diagnostic, les méthodes que nous jugeons pertinentes pour le diagnostic des Systèmes à Evénements Discrets (SED), et plus particulièrement pour le diagnostic des systèmes manufacturiers, autour desquels sera développé le chapitre 2.

1.2. Système automatisé de production

L'automatisation industrielle a connu, au cours de ces dernières décennies, une évolution importante consécutive à l'accroissement des exigences de qualité, de flexibilité et de disponibilité dans les procédés industriels [PER 04]. L'automatisation de ces derniers concerne tous les aspects de l'activité industrielle : production, assemblage, montage, contrôle, conditionnement, manutention, stockage, ... Son objectif est de réaliser, de manière automatique, des fonctions particulières répondant à des besoins spécifiques. Un Système Automatisé de Production (SAP) doit donc traiter une matière d'œuvre pour lui apporter une valeur ajoutée de façon reproductible et rentable.

1.2.1. Objectif de l'automatisation de production

Les productions industrielles sont de plus en plus automatisées [DEV 00]. Cette progression du degré de l'automatisation concerne l'automatisation d'opérations autrefois manuelles comme les assemblages ou les contrôles, mais aussi l'automatisation plus poussée d'opérations déjà partiellement automatisées. Ceci se trouve, par exemple, dans le passage en automatique de machines semi-automatiques ou le remplacement de machines rigides (ne fabriquant qu'un seul type de produit) par des machines flexibles susceptibles d'opérer sur plusieurs variantes de produits.

Les objectifs poursuivis par une automatisation peuvent être assez variés [PER 04] :

- Recherche de diminution du coût pour le produit, par réduction des frais de main-d'œuvre, d'économie de matière, d'économie d'énergie, ...,
- Suppression des travaux dangereux ou pénibles, et amélioration des conditions de travail par l'ennoblissement des tâches, ...,
- Recherche d'une meilleure qualité du produit, en limitant le facteur humain, et en multipliant les contrôles automatisés, ...,
- Réalisation d'opérations impossibles à contrôler manuellement ou intellectuellement, par exemple des assemblages miniatures, des opérations très rapides, des coordinations complexes, ...

1.2.2. Structure des systèmes automatisés

Le SAP se décompose en deux parties (Figure 1-1) : la Partie Opérative (PO) dont les actionneurs agissent sur le processus automatisé, et la Partie Commande (PC) qui coordonne les actions de la PO [PER 04].

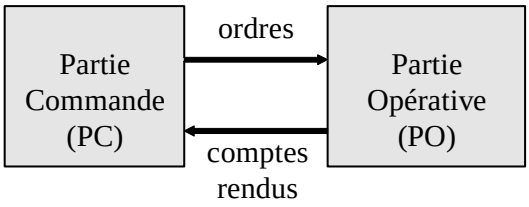


Figure 1-1 : Structure d'un SAP

La PO assure les transformations des produits entrants permettant d'élaborer la valeur ajoutée recherchée. C'est, en général, un ensemble mécanisé. Les principales technologies des chaînes d'action utilisent les énergies générées par de l'électricité, de l'air comprimé (pneumatique) et des fluides hydrauliques. Au plan informationnel, elle émet vers la PC des comptes-rendus caractéristiques des modifications opérées sur les produits. Elle reçoit des ordres qui sont des informations à caractère énergétique déclenchant, arrêtant ou modulant les transformations énergétiques nécessaires aux actions sur les matières d'œuvre (Figure 1-2a).

La PC reproduit le savoir-faire des concepteurs pour obtenir la suite des actions à effectuer sur les produits afin d'assurer la valeur ajoutée désirée. Elle commande la PO pour obtenir les effets voulus, par l'émission d'ordres en fonction d'informations disponibles, comptes-rendus, consignes et modèle de fonctionnement. Elle peut échanger des informations avec des humains ou d'autres systèmes (Figure 1-2b). La PC est construite à partir de constituants électroniques et électriques et s'appuie essentiellement sur des technologies programmées (automates programmables, microcontrôleurs). Les traitements sont réalisés au plus près des actions à réaliser sur les produits.

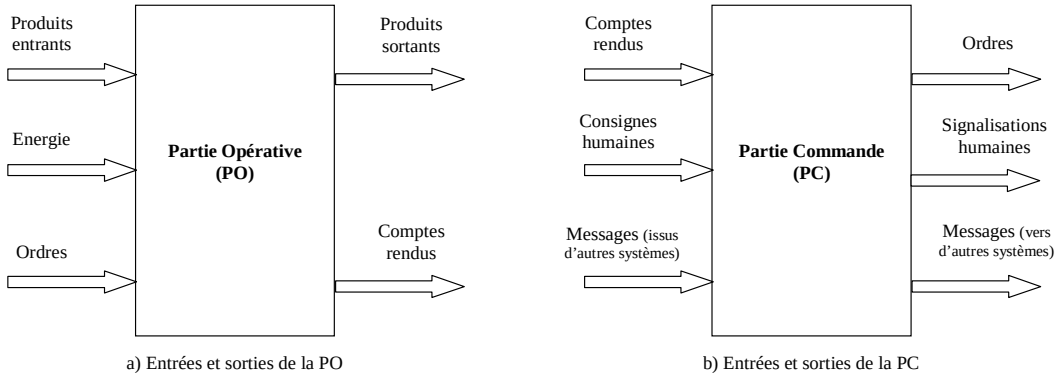


Figure 1-2 : Entrées et sorties de a) la PO et b) la PC

1.2.3. Eléments de la Partie Opérative

La Partie Opérative (PO) est une reproduction physique d'un procédé industriel [DEV 00]. Elle est soumise aux instructions envoyées de la commande par les Opérateurs Humains de Supervision (OHS) afin d'agir sur les actionneurs et pré actionneurs du procédé. En contre partie, la PO renvoie les informations du procédé aux OHS via des capteurs.

Trois technologies d'actionneurs se complètent pour répondre aux besoins des machines :

- les actionneurs électriques : moteurs, électrovannes, résistances de chauffage, électroaimants, ... Les préactionneurs associés à ces actionneurs électriques sont principalement les contacteurs, les variateurs de vitesse entourés des sécurités nécessaires, ….
- les actionneurs pneumatiques : vérins pneumatiques pour les mouvements de transfert, serrages, marquages, maintiens, assemblages, ... Les distributeurs sont les préactionneurs qui leur sont associés. Ils reçoivent un signal pneumatique électrique voire mécanique.
- les actionneurs hydrauliques : vérins ou moteurs. Ils ne sont utilisés que lorsque les actionneurs précédents ne peuvent donner satisfaction, lorsque les efforts à développer sont très importants ou lorsque des vitesses lentes doivent être contrôlées avec précision.

Les capteurs fournissent les informations en retour nécessaires pour la conduite du procédé en captant les déplacements des actionneurs ou le résultat de leurs actions sur le procédé. Ils peuvent détecter des positions, des pressions, des températures, des débits, des codes, des contraintes, des vitesses, ...

Le diagnostic va nécessiter une modélisation de la PC et de la PO. Le lecteur pourra retrouver dans l'annexe 1 différents outils de modélisation pour les systèmes manufacturiers.

1.3. Diagnostic des défaillances

Les SAPs représentent une catégorie importante des procédés industriels et introduisent une complexité d'exploitation en raison de l'imbrication d'éléments complexes de la PO.

Cette constitution rend l'analyse plus difficile et le taux d'apparition de pannes plus important. Les procédés doivent donc être suivis par des systèmes de surveillance afin de pouvoir alerter l'opérateur d'une panne et remonter l'information du diagnostic établi afin que l'OHS puisse prendre une décision via la supervision.

1.3.1. Surveillance, diagnostic et supervision

La surveillance des procédés industriels consiste à générer des alarmes à partir des informations délivrées par des capteurs [VAL 89]. Elle recueille les signaux en provenance du procédé et de la commande et reconstitue l'état réel du système commandé. Des seuils sont définis sur des variables clés par des experts du procédé selon des critères de sécurité concernant les hommes, l'installation et son environnement. Cette génération d'alarmes apporte une aide aux OHS dans leur tâche de surveillance afin qu'ils puissent analyser la situation et prendre une décision adaptée (procédure d'arrêt d'urgence, mode dégradé, action corrective). Elle a un rôle passif vis-à-vis du système de commande et du procédé [COM 91]. Cependant, la complexité et la taille de l'installation augmentent rapidement la quantité d'informations à analyser, rendant la surveillance complexe pour les OHS. Il est donc très utile d'adjoindre à la surveillance, une aide à la décision à travers un module de diagnostic.

Le diagnostic s'intègre dans le cadre plus général de la surveillance et de la supervision. C'est un système d'aide à la décision, son objectif est de localiser les composants ou les organes défaillants d'un procédé et éventuellement de déterminer les causes. Le diagnostic établit donc un lien de cause à effet entre un symptôme observé et la défaillance qui est survenue, tout en considérant qu'un même symptôme peut apparaître pour différentes causes [COM 00a].

La supervision a pour objectif de surveiller et de contrôler l'exécution d'une opération et le fonctionnement d'une installation. Elle a donc un rôle décisionnel et opérationnel en vue de la reprise de la commande. Elle est essentiellement effectuée par les OHS en salle de contrôle. La supervision élabore des solutions correctives en ayant la connaissance des causes, ou des organes ayant générés une défaillance.

1.3.2. Place de la détection

La définition établie ici pour le diagnostic, intègre le module de détection. En fait, cette fonction représente très souvent un sujet de débat concernant sa place précise. En effet, de nombreuses approches considèrent la détection comme un élément à part du diagnostic et le voient plutôt comme une entité de la surveillance [COM 91], [DAN 97], [COM 00b], [BOU 03a].

D'autres travaux préfèrent la considérer comme une information primordiale et indissociable du diagnostic et définissent le diagnostic comme la **détection**, la **localisation** et l'**identification** de défauts. Ce sont les méthodes à base de modèles appelées FDI (Fault Detection and Isolation) [DAR 03]. La détection permet de détecter tout écart du comportement normal du système et alerte les OHS de la présence d'un défaut. La localisation permet de remonter à l'origine de l'anomalie et de localiser le ou les composants défectueux. Cette localisation est importante puisque la propagation d'une panne provoque souvent l'apparition de nouveaux défauts. Enfin, l'identification détermine l'instant d'apparition de la panne, sa durée et son importance. Le diagnostic aide donc les OHS à surveiller un procédé

complexe et par conséquent à prendre une décision pour effectuer une reprise de la commande.

Par la suite, nos travaux s'articuleront autour d'une définition du diagnostic à base de modèles (FDI) comportant les trois activités : détection, localisation et identification des défauts (Figure 1-3).

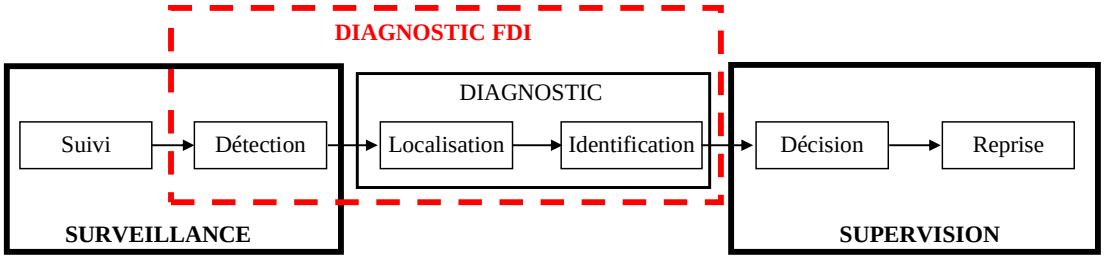


Figure 1-3 : Place de la détection dans le diagnostic FDI

1.3.3. Terminologie du diagnostic

Un défaut (Fault) est considéré comme un écart du comportement normal. Cet écart est un dysfonctionnement qui n'empêche pas un procédé à remplir sa fonction. Il s'exprime par une déviation d'une propriété ou d'un paramètre caractéristique du procédé. Un défaut est donc une anomalie de comportement qui peut présager d'une défaillance à venir [TOU 05]. Il peut trouver son origine dans les classes suivantes : défaut d'un composant (actionneur ou capteur), défaut de la commande ou défaut dû à une faute d'un OHS.

Une défaillance (Failure) est une anomalie fonctionnelle qui empêche partiellement l'aptitude d'un procédé à remplir sa fonction. Une défaillance est donc inclue dans les défauts.

Une panne (Break-down) représente les conséquences d'une défaillance dans la réalisation du fonctionnement nominal du procédé [ZWI 95]. Elle provoque un arrêt complet du procédé [TOU 05]. Une panne peut être considérée comme permanente ou intermittente :

- **Les pannes permanentes** sont définies comme un mauvais fonctionnement d'un composant qui doit être changé ou réparé. Elles peuvent être la conséquence du changement progressif des caractéristiques d'un composant, comme le vieillissement par exemple, ou un changement brutal comme une casse matériel.
- **Les pannes intermittentes** peuvent, quant à elles, permettre un retour du procédé dans sa dynamique de fonctionnement. Par exemple, une canalisation bouchée peut être débouchée par pression interne. Ces pannes sont très souvent le prélude à une panne permanente et expriment une dégradation progressive des performances du procédé.

Nous utilisons le terme défaut dans ce mémoire de thèse puisqu'il inclut la défaillance et la panne.

Le diagnostic prédictif (Prognostic) peut être défini comme l'anticipation d'une panne due à une dégradation progressive. Il est donc intéressant puisqu'il permet une intervention préventive sur le procédé pour éviter les conséquences d'une panne qui peuvent être catastrophiques comme dans le cas des centrales nucléaires.

1.3.4. Caractéristiques des méthodes de diagnostic

Il existe un grand nombre de méthodes de diagnostic [BOU 96], [BRU 90], [COM 91], [DEB 00], [DUB 90], [DUB 01], [FRA 90], [GAR 05], [GEN 03], [HOL 94], [LIN 94], [PAN 00], [PHI 05], [SAM 01], [MOU 02], [SCH 87], [SRI 93], [SU 04], [WAN 00]... Ces méthodes se basent sur un modèle du comportement normal et/ou défaillant du système. L'observation réelle de l'état courant du système, sujet du diagnostic, est comparée avec l'état estimé par le modèle afin de détecter un défaut. Chacune des méthodes de diagnostic doit garantir les caractéristiques suivantes :

- Le diagnostic doit être facile à implémenter,
- Le nombre de capteurs nécessaire pour le diagnostic doit être minimal,
- Le diagnostic doit être prédictif,
- Le diagnostic doit être réalisable en temps réel,
- Le diagnostic doit être concevable algorithmiquement.

1.4. Classification des méthodes de diagnostic

Les méthodes de diagnostic se distinguent selon différents critères : la dynamique du procédé (discret, continu ou hybride), sa complexité, l'implémentation du diagnostic en ligne et/ou hors ligne, la nature de l'information (qualitative et/ou quantitative), sa profondeur (structurelle, fonctionnelle et/ou temporelle), sa distribution (centralisée, décentralisée ou distribuée), …. En général, ces méthodes sont divisées en deux catégories :

- Les méthodes externes (model-free methods) qui sont des méthodes soit à base de connaissances, soit des méthodes empiriques et/ou de traitement du signal.
- Les méthodes internes (model-based methods) qui représentent des méthodes à base de modèles quantitatifs et/ou qualitatifs.

Nous présentons rapidement dans la suite, une synthèse des méthodes de diagnostic selon cette classification en deux catégories (interne et externe). L'objectif est de positionner la catégorie des méthodes de diagnostic qui nous intéresse pour notre étude parmi les différentes méthodes de diagnostic de la littérature.

1.4.1. Méthodes externes

Les méthodes externes considèrent le système comme une "boîte noire" et elles n'ont besoin d'aucun modèle mathématique pour représenter le fonctionnement du procédé. Elles utilisent uniquement un ensemble de mesures et/ou de connaissances heuristiques sur le système. Ces méthodes comprennent les méthodes à base de systèmes experts [ZWI 95], de reconnaissance des formes [DUB 01], [MOU 02], [ANI 00] ou de réseaux de neurones [ZWI 95], [DUB 01].

Les systèmes experts utilisent une information heuristique pour lier les symptômes aux

défauts. Ce sont des systèmes à base de règles qui établissent des associations empiriques entre effets et causes. Ces associations sont généralement fondées sur l'expérience de l'expert plutôt que sur une connaissance de la structure et/ou du comportement du système. Ils font partie alors des systèmes dits *à connaissance de surface*. Leur fonctionnalité est de trouver la cause de ce qui a été observé en parcourant les règles par des techniques classiques :

- Le raisonnement inductif par chaînage avant, afin de trouver tous les symptômes qui sont la conséquence d'un symptôme initial (état réel du système),
- Le raisonnement déductif par chaînage arrière, afin de trouver toutes les causes possibles qui peuvent expliquer un symptôme.

La qualité première des systèmes experts réside dans leur efficacité au niveau temps de calcul. Le système doit simplement attendre les événements observables des règles pour "sauter" directement aux conclusions [UNG 93]. De plus, l'interprétation du résultat est généralement compréhensible pour l'opérateur puisque ces règles sont le produit d'experts humains. Enfin, les systèmes experts sont facilement implantables puisqu'il s'agit d'une énumération de règles. Cependant, ils sont totalement dépendants de l'expertise, et les règles acquises sur une application ne peuvent pas être utilisées sur une autre application. Dans le cas de nouveaux systèmes, il n'y a pas ou peu d'expériences au sujet des pannes pouvant se produire, ce qui rend difficile l'acquisition des règles. Un système expert ne peut être opérationnel dès le début de son exploitation et a donc besoin de temps pour son apprentissage. Il en est de même lors d'ajouts ou de suppressions de composants. Enfin, un système expert est la conséquence d'une règle reconnue. Il ne donne pas d'explication sur les conclusions données et rend donc difficile la détection sur la propagation de pannes.

La reconnaissance des formes se base sur la définition d'algorithmes permettant de classer des objets ou des formes en les comparant à des formes types. Une forme est l'observation du fonctionnement du procédé, représentée par un point dans un espace à *n* paramètres appelé "espace de représentation". Les formes types sont les modes de fonctionnement représentés par des ensembles de points occupant des zones restreintes de l'espace de représentation appelées classes. Le diagnostic par reconnaissance des formes est réalisé en utilisant une méthode de classification qui associe une nouvelle observation à une classe correspondant à un mode de fonctionnement.

L'utilisation de la reconnaissance des formes en diagnostic est surtout efficace en terme de temps de calcul pour la classification d'une nouvelle observation, et de capacité à traiter des données qui sont à la fois incertaines et imprécises. Elle est également capable de réaliser un diagnostic prédictif et de travailler avec des bases de données incomplètes. Cependant, l'inconvénient majeur de l'utilisation de la reconnaissance des formes en diagnostic réside dans la difficulté à trouver l'ensemble minimal mais suffisant des paramètres informatifs modélisant le fonctionnement réel du procédé. Toutefois, le diagnostic par reconnaissance des formes a été appliqué avec succès sur différentes applications réelles [ANI 00], [DUD 01], [MOU 02].

1.4.2. Méthodes internes

Dans la plupart des systèmes automatisés, la PC d'un procédé est généralement représentée à travers un modèle devant être appliqué sur la PO. Pour réaliser un diagnostic, il faut également pouvoir représenter l'état de la PO à travers un modèle qui peut être intégré au

modèle de commande, séparé ou mixte. Ainsi, lorsqu'un défaut apparaît, il est possible de disposer d'informations concernant le procédé et de comparer modèle et procédé. On parle alors de diagnostic à base de modèles (Fault Detection and Isolation) [DAR 03]. Le diagnostic à base de modèles génère des indicateurs de défauts, résidus, contenant des informations sur les anomalies ou les dysfonctionnements du procédé à diagnostiquer. Un écart entre l'état réel de la PO et celui estimé par le modèle, représentant le fonctionnement nominal, est mesuré. Les résidus doivent alors être assez sensibles aux défauts pour leur détection, localisation et identification (Figure 1-4).

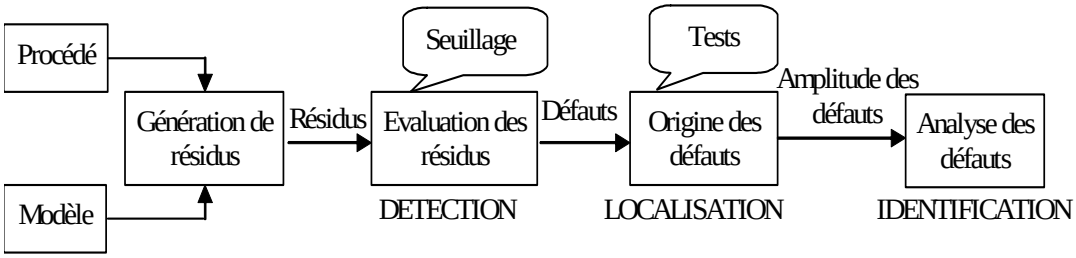


Figure 1-4 : Diagnostic à base de modèles

Parmi les méthodes internes à base de modèles, on peut distinguer les méthodes basées sur des modèles quantitatifs, les méthodes basées sur des modèles qualitatifs et les méthodes basées sur les deux modèles.

Les modèles quantitatifs sont utilisés pour l'estimation de paramètres, d'état ou d'espace de parité à travers des modèles mathématiques et/ou structurels pour représenter l'information disponible du fonctionnement d'un procédé. Un défaut provoque alors des changements dans certains paramètres physiques du procédé. Les modèles mathématiques comparent les différentes valeurs des variables avec des seuils de détection afin de générer un résidu qui sera fourni au diagnostic. A partir de toutes les signatures de défauts connues par apprentissage, il est possible d'isoler et d'identifier la panne avant de prendre une décision [ZWI 95] [BRU 90] [FRA 90]. Les avantages de ces méthodes internes sont tout d'abord la capacité à détecter les variations abruptes et progressives de pannes à travers une analyse des tendances des signaux. De plus, ces méthodes possèdent la capacité de donner une localisation précise du défaut. Par contre, elles nécessitent une information dite "profonde" sur le comportement du système et de ses pannes, rendant les calculs complexes pour le diagnostic en ligne. Elles sont également très sensibles aux erreurs de modélisation.

Les méthodes à base de modèles qualitatifs permettent de représenter le comportement du procédé avec un certain degré d'abstraction à travers des modèles non plus mathématiques mais des modèles de type symbolique [TRA 97]. Les modèles qualitatifs doivent représenter de manière qualitative des systèmes continus, discrets et/ou hybrides pour que le diagnostic soit capable de détecter les déviations du fonctionnement normal, localiser la défaillance et en déterminer la ou les causes. Pour les systèmes continus, les modèles qualitatifs sont fréquemment basés sur des graphes causaux [BOU 98] et [MON 00] ou des graphes causaux temporels [MOS 01]. Une abstraction qualitative des comportements continus peut être représentée par des modèles à base d'événements discrets (SED) [KOU 00] et [SU 03], ou la théorie de supervision [RAM 89a]. Pour les SED, de nombreuses approches sont proposées utilisant des outils tels que les automates, les équations logiques ou les RdP avec observation partielle ou totale du fonctionnement du procédé. Ces approches seront traitées en détail dans le chapitre 2. Enfin, une intégration des modèles discrets et des modèles continus peut être retrouvée également dans les systèmes dynamiques hybrides [ALL 98] et [KOU 98].

Les méthodes à base de modèles quantitatifs et qualitatifs reposent, d'une part, sur une évaluation quantitative pour la détection d'un défaut, et d'autre part sur une analyse qualitative des transitoires pour la localisation et l'identification. Ces méthodes ont l'avantage de combiner les points forts des méthodes à base de modèles quantitatifs et à base de modèles qualitatifs. Cependant, elles sont lourdes à implémenter. On peut citer comme exemple de ces méthodes, celle développée par [MAN 00].

1.4.2.1. Approches d'implantation du diagnostic

Différentes approches sont employées dans la littérature pour l'implantation d'un module de diagnostic. Les approches les plus utilisées sont : l'approche comparateur, l'approche référence et l'approche filtre.

L'approche "comparateur" utilise un émulateur qui contient le modèle des évolutions normales du procédé (Figure 1-5). La détection s'effectue en comparant les signaux réellement émis par la PO avec ceux provenant de l'émulateur dans une fenêtre temporelle pendant laquelle devront apparaître les différents comptes-rendus. Une incohérence est détectée par le comparateur quand il y a une apparition d'un compte-rendu non attendu ou lorsqu'il y a non apparition d'un compte-rendu pendant la fenêtre temporelle correspondante. Un exemple de l'utilisation de cette approche peut être trouvé dans [HOL 90] et [TOG 92]. L'inconvénient de cette approche est l'envoi de l'ordre en même temps à la PO et à l'émulateur. Cela signifie qu'un ordre erroné ou qui ne correspond pas à l'état actuel de la PO, ne sera pas validé avant son application sur la PO.

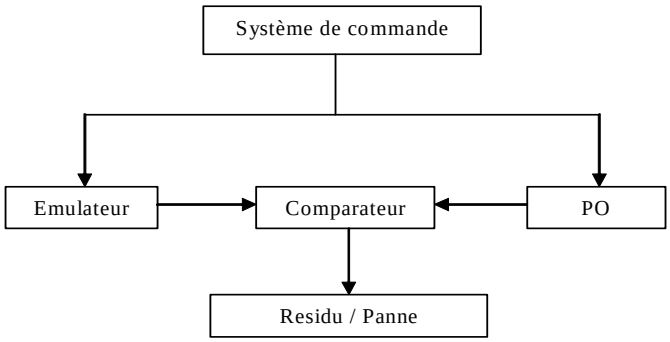


Figure 1-5 : Approche "comparateur"

Dans [COM 91], un modèle de référence contenant le modèle des comportements normaux du procédé est présenté. Le principe est de consulter le modèle de référence avant l'envoi de requêtes dans une fenêtre temporelle (Figure 1-6). S'il y a incohérence entre la requête envoyée et l'état du modèle de référence, alors une erreur de la PC est détectée. Les 2 modèles doivent évoluer simultanément, sinon seule la référence évolue et une défaillance de la PO est détectée. L'avantage de cette approche est qu'un ordre ne sera pas envoyé à la PO avant de le valider. Cependant, les comptes-rendus émis par la PO ne sont pas, à leur tour, validés avant de les envoyer vers la PC.

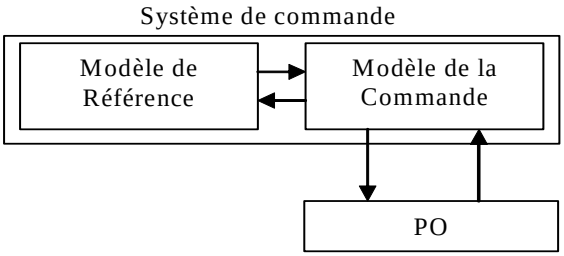


Figure 1-6 : Approche par modèle de référence

Dans [LHO 91] ou [CRU 91], le modèle de la PO est placé dans un filtre qui est intercalé entre le système de commande et la PO afin de valider chaque envoi d'ordres ou de valeurs capteur (Figure 1-7). L'envoi d'un ordre par la PC passe par un filtre de commande qui teste la cohérence de ce dernier par rapport à la situation de la PO avant de le valider. Inversement, la PO renvoie la valeur de ses observations à un filtre de capteurs avant de les valider. Cette approche a l'avantage d'empêcher la propagation d'un défaut en le détectant au plus tôt. Cependant, elle demande une connaissance profonde de la PO mais aussi de la PC autour de modèles riches. C'est cette approche que nous adapterons pour l'implantation du module de diagnostic proposé au chapitre 3.

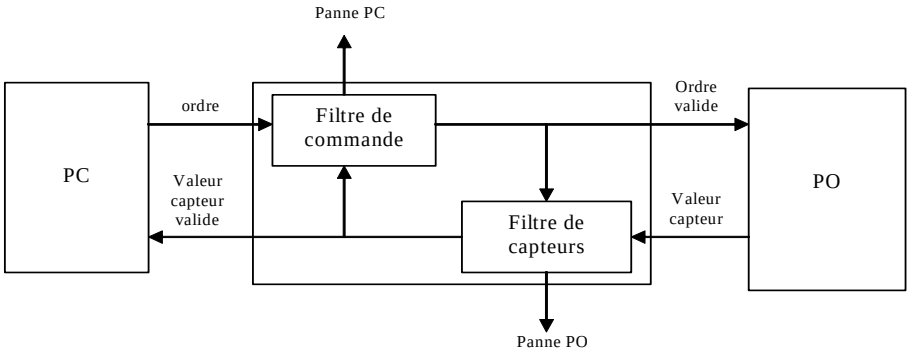


Figure 1-7 : Approche Filtre

1.4.2.2. Structure de la prise de décision du diagnostic

Une structure pour la prise de décision du diagnostic doit être définie pour les modèles qualitatifs. En effet, pour ces modèles, l'information est très souvent représentée par des outils symboliques s'exposant à des problèmes d'explosion combinatoire et/ou de communication entre les différents composants d'un procédé. Le choix d'une structure dépend de la distribution de l'information disponible : centralisée ou distribuée, et de la taille du procédé : simple ou complexe. Il existe donc trois grandes structures de prise de décision des méthodes de diagnostic : centralisée, décentralisée et distribuée. Elles sont présentées brièvement dans ce paragraphe et seront approfondies dans le chapitre 2.

La structure centralisée consiste à associer un modèle global du procédé avec un seul module de diagnostic, que nous appelons "diagnostiqueur" (Figure 1-8). Ce dernier collecte les différentes informations du procédé avant de prendre sa décision finale sur l'état de fonctionnement du procédé [SAM 94]. Bien que performantes en terme de diagnostic, la structure centralisée est difficilement utilisable pour les systèmes de grande taille. En effet, la constitution d'un modèle global du procédé engendre très souvent des problèmes d'explosion combinatoire.

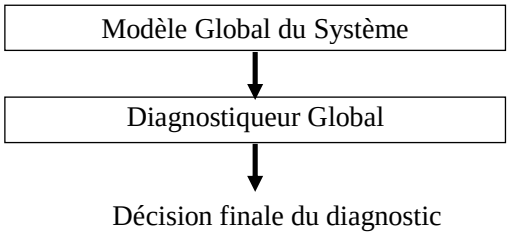


Figure 1-8 : Structure Centralisée

La structure décentralisée se base sur un modèle global du procédé à qui sont associés plusieurs diagnostiqueurs locaux indépendants (Figure 1-9). Chaque diagnostiqueur reçoit les observations qui lui sont spécifiques et prend une décision locale en se basant sur ses observations locales [CHA 93]. Cependant, cette structure implique des problèmes d'indécisions. En effet, certaines spécifications globales ne peuvent pas être représenté par un diagnostiqueur local. Afin de résoudre le cas d'indécision, chaque diagnostiqueur envoie sa décision locale à un coordinateur (ou superviseur) qui va gérer les différents problèmes d'ambiguïté entre les diagnostiqueurs et va prendre la décision finale [DEB 00], [WAN 04]. Nous adapterons cette structure pour notre module de diagnostic développé en chapitre 3 puisqu'elle est adaptée aux systèmes manufacturiers.

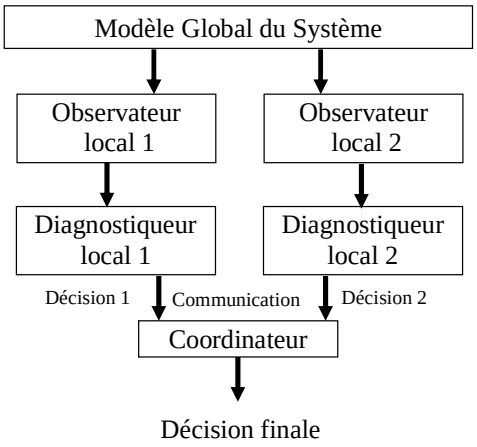


Figure 1-9 : Structure Décentralisée, cas de 2 diagnostiqueurs locaux

Dans la structure distribuée, le procédé est modélisé à travers ses composants par plusieurs modèles locaux. Chacun étant associé à un diagnostiqueur local responsable de son composant (Figure 1-10). Dans le cas de spécifications globales, un protocole de communication permet la communication directement entre les différents diagnostiqueurs afin de gérer les conflits décisionnels [DAS 03a], [SU 04] et [QIU 05]. Chaque diagnostiqueur prend sa décision en se basant sur sa propre observation locale et celle communiquée par les autres diagnostiqueurs locaux. Cette structure permet donc de s'affranchir de la construction d'un coordinateur mais implique une définition d'un protocole de communication entre les diagnostiqueurs parfois difficilement réalisable et engendrant des délais de communication plus important.

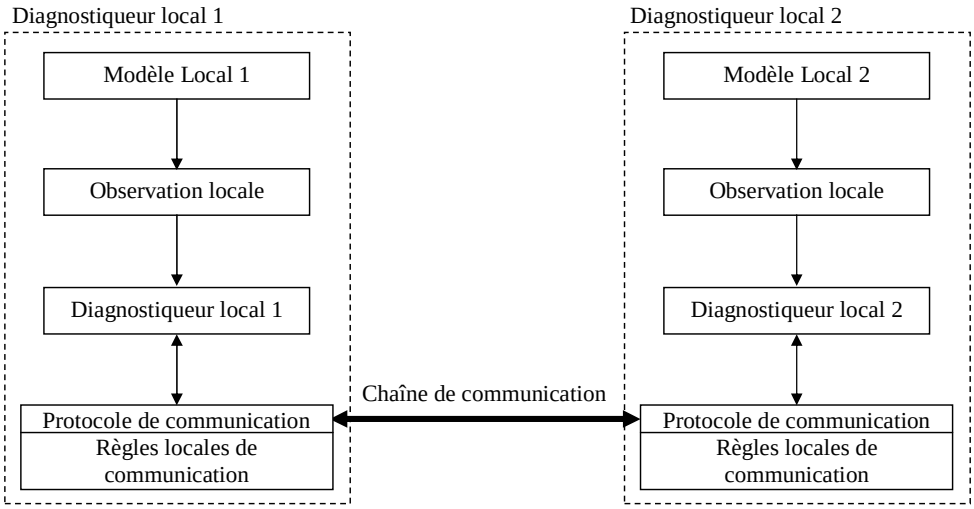


Figure 1-10 : Structure Distribuée, cas de 2 diagnostiqueurs locaux

1.4.2.3. Approches de construction d'un module de diagnostic

Il existe trois grandes approches de construction d'un module de diagnostic : l'approche intégrée, l'approche séparée et l'approche mixte. La différence est liée à la place du module de diagnostic par rapport au système de commande [SAH 87], [COM 01].

Pour l'approche intégrée, le diagnostic est pris en compte dès la conception du système de commande et y intègre les états défaillants. Les situations normales et anormales sont gérées par le système de commande. Un seul outil de spécification est alors nécessaire pour la mise en place de cette approche. Cependant, une liste exhaustive des situations anormales doit être connue pour être intégrée au système de commande. Elle permet de réagir rapidement lors de l'occurrence d'une défaillance, notamment pour les procédés simples où le nombre d'états est faible. On retrouve des méthodes de synthèse de commande visant à interdire les événements conduisant à des fonctionnements dangereux à partir d'un modèle de procédé et de commande [RAM 89a]. L'approche intégrée a tendance à limiter le problème d'explosion combinatoire car elle exprime uniquement le cahier des charges du procédé.

L'approche séparée considère le diagnostic comme une fonction à part entière. La commande ne traite alors aucun aspect du traitement des défaillances et le module de diagnostic doit y être ajouté. Cette séparation a l'avantage de pouvoir utiliser des modèles, des techniques et des outils plus adaptés et plus efficaces pour le diagnostic que ceux utilisés pour la commande. Par exemple, les techniques de l'Intelligence Artificielle permettent de mettre en œuvre un diagnostic pour des procédés mal connus où les données sont incertaines, imprécises et incomplètes. Cependant, cette approche présente l'inconvénient majeur de générer des conflits entre le diagnostic et la commande puisqu'ils agissent tous les deux sur le procédé. Ces conflits proviennent de la séparation entre les situations normales et anormales. En effet, ce qui est normal pour la surveillance ne l'est pas forcément pour la commande. On retrouve cette approche utilisée par exemple dans les travaux de [HOL 91] et [TOG 92].

L'approche mixte se trouve à mi-chemin entre les approches intégrée et séparée. Elle est particulièrement adaptée pour les procédés manufacturiers car elle utilise une détection intégrée qui s'adapte aux séquences de commande en cours d'exécution avec un module de diagnostic séparé utilisant les techniques de raisonnement les mieux adaptées [COM 01]. [COM 98] et [ZAM 98] ont présenté une approche de commande et de supervision permettant

de prendre en compte les différentes fonctions du diagnostic. Le diagnostic est alors vu comme une approche mixte puisque la commande est construite de manière indépendante du raisonnement de l'approche tout en intervenant à travers ses séquences pour diriger le processus de diagnostic. C'est cette approche que nous choisirons pour notre module de diagnostic développé au chapitre 3.

1.5. Conclusion

L'évolution des Systèmes Automatisés de Production (SAP) rend le diagnostic des défaillances indispensable pour la production et le développement industriel. Les différents éléments de la Partie Opérative (PO) que l'on retrouve dans les milieux industriels composent une source d'informations primordiale au diagnostic. Ce chapitre a présenté, dans un premier temps, notre vision du diagnostic des défaillances par rapport à la surveillance et la supervision. En effet, nous considérons le diagnostic FDI (Fault Detection and Isolation) comme une composition de trois modules : la détection, la localisation et l'identification. Dès lors, une terminologie appropriée à notre vision du diagnostic a été présentée afin de définir et de délimiter chaque terme et chaque notion utilisés dans ce mémoire de thèse.

Ensuite, une première étude de classification des différentes méthodes de diagnostic a donc été présentée dans ce chapitre. Deux grandes catégories de méthodes de diagnostic ont été dégagées : méthodes externes et internes. Les méthodes externes sont essentiellement basées sur les connaissances de l'expert. Les méthodes internes sont, quant à elles, représentées par des modèles quantitatifs et/ou qualitatifs. Les méthodes internes à base de modèles qualitatifs permettent une modélisation du procédé et de la commande. Ils sont par conséquent très représentatifs des Systèmes à Evénements Discrets (SED) autour desquels ce mémoire de thèse est centré. Une présentation des approches d'implantation, les différentes structures de la prise de décision du diagnostic et les approches de construction d'un module du diagnostic ont été introduites. La Figure 1-11 résume l'ensemble de la classification des approches de diagnostic des défaillances.

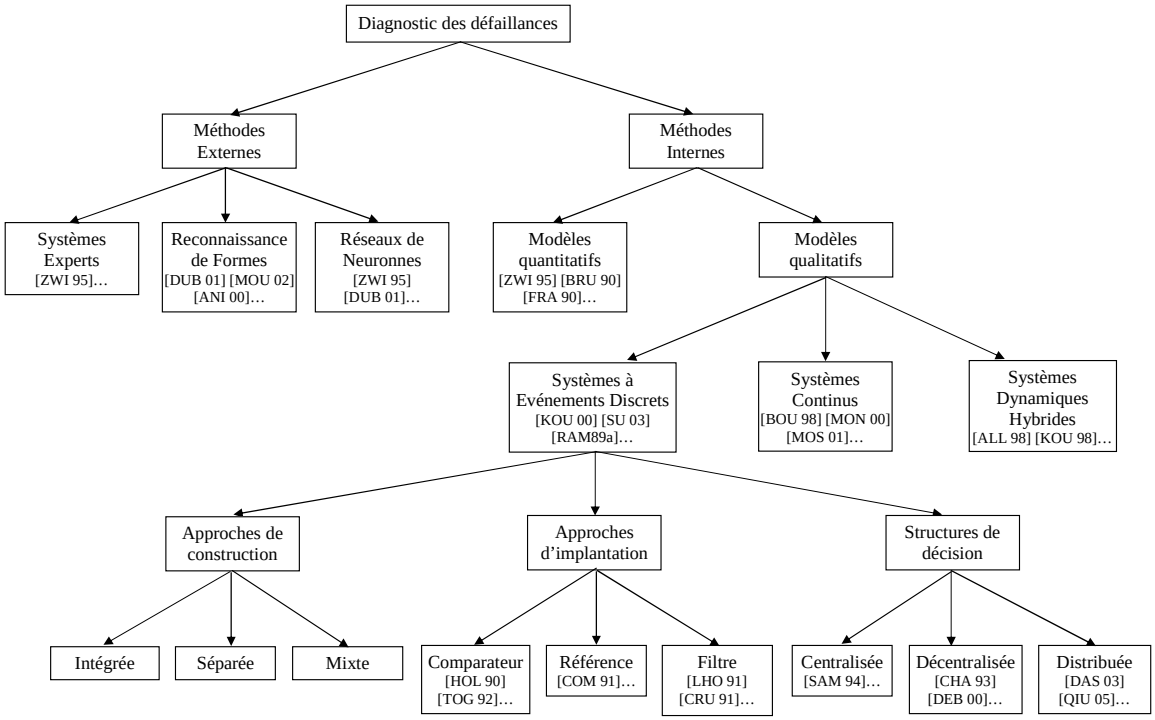


Figure 1-11 : Classification des approches de diagnostic des défaillances

Dans le chapitre 2, nous orienterons nos propos vers les méthodes de diagnostic des SED, et plus particulièrement vers les systèmes manufacturiers. Une classification et une analyse approfondie de ces méthodes seront présentées. Nous développerons également une présentation plus détaillée des structures de la prise de décision du diagnostic des SED ainsi qu'une introduction aux différentes formulations du problème de diagnostic des SED. Nous verrons que les systèmes manufacturiers sont des systèmes informationnellement décentralisés et modulaires. C'est pourquoi nous nous intéresserons plus particulièrement à la structure décentralisée. Cet état de l'art permettra alors de situer l'approche que nous allons proposer en chapitre 3 parmi les autres méthodes de diagnostic des SED proposées dans la littérature.

Chapitre 2 : Diagnostic des Systèmes à Evénements Discrets : Contexte et problématique

2.1. Introduction et problématique du diagnostic des SED

Le diagnostic des défaillances est indispensable à la bonne conduite d'un Système Automatisé de Production. Ce deuxième chapitre présente un état de l'art des méthodes de diagnostic des défaillances appliquées aux Systèmes à Evénements Discrets (SED). Le lecteur trouvera en Annexe 1 toute une partie consacrée aux SED, à leurs applications, aux outils de modélisation et les notations correspondantes.

Dans le cadre de notre étude, nous souhaitons orienter nos travaux plus particulièrement vers les systèmes manufacturiers qui représentent une catégorie importante des SED et qui interviennent sur les procédés de production, de traitement et de transformation de produit. Les systèmes manufacturiers sont constitués de différents éléments technologiques devant réagir entre eux et constituant ce qu'on appelle la Partie Opérative PO. On parle alors de systèmes informationnellement décentralisés sur chaque élément.

La problématique du diagnostic des SED est essentiellement liée au fait que les modèles représentant les SED ne sont pas riches en terme d'informations. Par conséquent, il faut exploiter toute l'information disponible sur le procédé. Il s'agit de prendre en compte à la fois :

- Le comportement événementiel de la PO à travers toutes ses situations possibles. Son modèle suppose une connaissance précise de la technologie du matériel du point de vue de son architecture (composants, agencement, articulations, mécanismes de synchronisation et d'exécution). Il fait très souvent appel à l'expert à travers un modèle intuitif du système.
- Les spécification obtenus à partir d'un cahier des charges. L'ensemble de ces spécifications est écrit par un langage et traduites sous forme d'un modèle de commande utilisé par la PC.
- L'information temporelle qui représente la réactivité des actionneurs à un ordre envoyé par la PC. Elle peut être représentée par des contraintes temporelles, comme par exemple les chroniques ou les *template*s. Ces contraintes informent sur le temps de réponse à un ordre (temps de sortie d'un vérin, transfert d'un chariot, …) et donc fournissent une donnée importante pour le diagnostic qui n'est pas toujours fournies par la commande.
- Le modèle symbolique qui décrit les réactions particulières et les contraintes globales

du procédé qui ne sont inclues dans aucun des modèles précédents. Il doit analyser les situations ou comportements incohérents à partir d'une expertise, de documentation, du cahier des charges, d'historiques,…. Le modèle symbolique peut s'exprimer sous forme de règles algébrique ou sous forme d'automates.

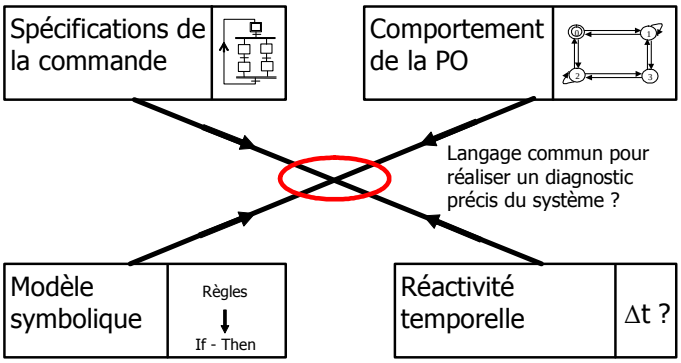


Figure 2-1 : Problématique du diagnostic des SED

Ces différentes informations doivent permettre de construire un ou des diagnostiqueur(s) représentant un modèle évolué riche du comportement du procédé, qu'il soit normal et/ou défaillant. La problématique, que nous posons pour réaliser un diagnostic des défaillances, consiste à faire cohabiter toutes ces informations (PC, PO, symbolique et réactivité temporelle) (Figure 2-1). Plusieurs outils de représentation sont alors nécessaires afin de modéliser ces informations.

Une seconde difficulté est la complexité des procédés manufacturiers qui entraîne l'explosion combinatoire. Egalement, ces procédés sont informationnellement décentralisés. Il faut donc une structure de la prise de décision adaptée. Nous présentons alors les différentes structures de la prise de décision, introduites brièvement en chapitre 1.

Ensuite et afin de vérifier si une approche de diagnostic est capable de diagnostiquer chaque défaut appartenant à l'ensemble des défauts prédéfinis, une notion de diagnosticabilité doit être définie. Nous étudions les différentes notions développées dans la littérature, en montrant leurs points forts et faibles.

Enfin, une discussion termine ce chapitre autour d'une étude comparative des méthodes de diagnostic des SED dans le but de justifier nos choix d'outil de représentation, de structure de la prise de décision et de notion de diagnosticabilité pour l'approche que nous proposerons dans le chapitre 3 pour le diagnostic des systèmes manufacturiers.

2.2. Méthodes de diagnostic des SED

La conception des méthodes de diagnostic des SED repose sur un certain nombre de critères touchant aux outils de représentation de la PC et de la PO, au type de modélisation des défauts, à la structure de la prise de décision des diagnostiqueurs et la notion de diagnosticabilité (Figure 2-2). Le diagnostic des SED nécessite la description complète du comportement du système autant du point de vue de la PC que de la PO. Nous allons voir maintenant l'ensemble de ces critères plus en détail.

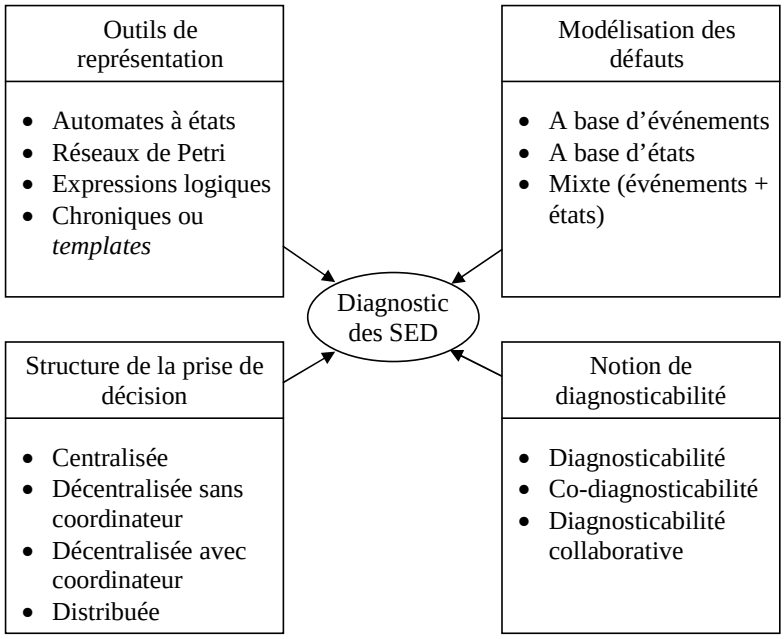


Figure 2-2 : Critères de classification des méthodes de diagnostic

2.2.1. Outils de représentation pour le diagnostic des SED

2.2.1.1. Diagnostic à base d'automates à état

L'automate est un outil adapté pour la modélisation de l'information sous forme d'événements observables et non observables par l'opérateur (voir Annexe 1). Ce modèle permet de décrire les évolutions à travers des séquences d'événements qui rendent compte de l'état du système. C'est un outil largement utilisé dans les méthodes de diagnostic des SED surtout lorsqu'il s'agit de décrire le comportement complet du système [SAM 95] [SU 04].

Prenons un exemple tiré de [SAM 95] représentant un procédé de distribution d'eau composé d'une vanne et d'une pompe. La modélisation du procédé comporte à la fois une description de la PO de la vanne et de la pompe (Figure 2-3a et Figure 2-3b), et un modèle de la commande (Figure 2-3c). Le modèle de la vanne intègre le comportement normal et anormal. Les états VF et VO représentent respectivement la vanne en position fermée et ouverte avec les événements d'ouverture OV et de fermeture de vanne FV. Ce modèle exprime également deux événements de défauts par les événements non observables FC et OC représentant respectivement un défaut sur la vanne en fermeture, VFC, et en ouverture, VOC, lorsqu'elle est collée.

La commande décrit le comportement désiré à partir de l'état initial C1 (Figure 2-3c). A partir de cet état, il est possible de demander une consommation d'eau par l'événement *conso* (état C2). Dans ce cas, la pompe doit démarrer par l'ordre DP (état C3) et ouvrir la vanne en activant l'ordre OV (état C4). Lorsque le consommateur arrête sa demande, alors l'événement *noconso* emmène le procédé dans l'état C5. Il faut alors fermer la vanne en envoyant l'ordre FV (état C6) puis arrêter la pompe par l'ordre AP (état C7).

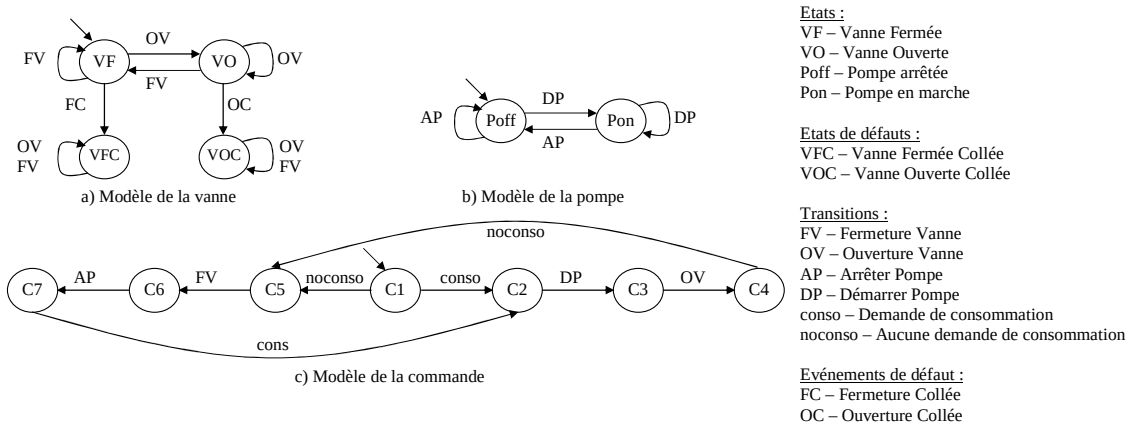


Figure 2-3 : Modèles de PO et PC dans un système de distribution d'eau

Le système est supposé n'avoir qu'un seul capteur de flux indiquant la présence d'un flux dans la canalisation par une mesure F et une non-présence de ce flux par une mesure NF. L'ensemble des indications de ce capteur pour ce système est répertorié dans le Tableau 2-1. Le capteur fournit une mesure en fonction de l'état de la vanne et de la pompe par une fonction H(Vanne, Pompe, ●) où le "●" précise que le capteur est totalement indépendant de l'état de la commande. Dès lors, on voit que lorsque la vanne est fermée et que la pompe est arrêtée, H(VF, Poff, ●), le capteur retourne la mesure de non flux NF.

H(VF, Poff, ●) = NF
H(VFC, Poff, ●) = NF
H(VOC, Poff, ●) = NF
H(VF, Pon, ●) = NF
H(VO, Pon, ●) = F
H(VFC, Pon, ●) = NF
H(VOC, Pon, ●) = F

Tableau 2-1 : Liste des mesures du capteur de flux pour le système de distribution d'eau

A partir des modèles de PO, de la commande et de la liste des mesures du capteur de flux, il est possible d'obtenir un modèle global de ce procédé par composition synchrone de tous les modèles (Figure 2-4). Dès lors, l'automate résultant décrit, à partir des états initiaux de chaque modèle, le comportement global du système. Ainsi, l'état 1 correspond à la composition de l'état C1 de la commande, de l'état VF de la vanne et de l'état Poff de la pompe. A partir de cet état, il est possible d'évoluer dans l'automate de commande par les événements de demande de consommation *conso* (état 3) où le capteur précise la non présence de flux dans la canalisation, dans l'état 4 s'il n'y a aucune demande de consommation ou dans l'état 2 puis 5 si un défaut sur la vanne collée est survenu par l'événement FC avant la demande de consommation. Pour détecter qu'un défaut est survenu, il faut pouvoir différencier l'état 3 et 5. Cette détection ne peut se faire qu'après un nouvel événement observable sur la vanne. A partir des états 3 et 5, il faut donc attendre tout d'abord le démarrage de la pompe par l'ordre DP pour arriver dans les états 7 et 9. C'est ensuite l'ordre d'ouverture OV qui va être associé soit à une mesure F du capteur si le procédé est en

fonctionnement normal pour aller à l'état 11, soit à une mesure NF indiquant que le flux n'est pas présent et qu'il y a donc un défaut (état 13). Ainsi, il est possible de différencier l'ensemble des états normaux correspondant à un défaut {1, 3, 4, 7, 8, 11, 12, 15, 19} des états anormaux dus à l'occurrence d'un défaut {2, 5, 6, 9, 13, 14, 16, 17, 18, 20, 21, 22}.

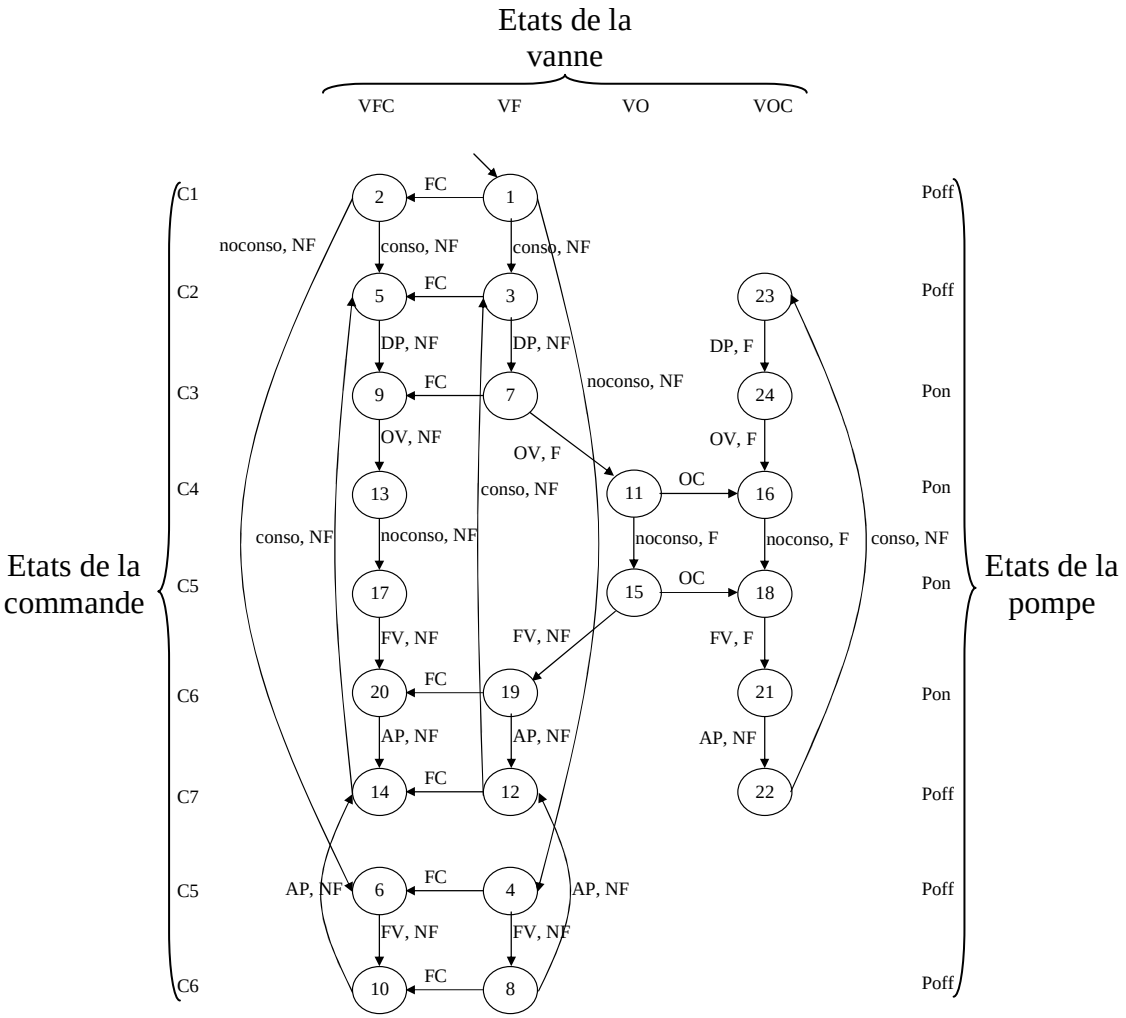


Figure 2-4 : Composition synchrone des différents modèles locaux

Cet exemple présente uniquement les défauts sur les actionneurs. Si l'on souhaite diagnostiquer les défauts sur les capteurs, il faut alors modéliser chaque capteur par un automate à état. Dès lors, la représentation d'un modèle global entraîne inévitablement une explosion combinatoire du nombre d'états.

Selon le degré de description, les automates à état modélisent les états et les séquences d'événements du système afin de fournir par la suite le maximum d'informations au diagnostiqueur qui, en fonction de cette information, prendra la décision finale sur la présence d'un défaut et sur son type (VFC ou VOC). Nous verrons un peu plus loin comment il est possible d'obtenir un diagnostiqueur à partir de ce modèle.

2.2.1.2. Diagnostic à base de Réseaux de Petri

Les Réseaux de Petri (RdPs) sont utilisés pour le diagnostic des SED dans de nombreuses approches. [VAL 89] décrit des systèmes manufacturiers par RdPs associés à une fonction

floue sur les transitions afin de définir des intervalles sur les délais d'exécution des ordres. [SRI 93] utilise le RdP temporel pour représenter le fonctionnement normal du système pour la détection. Associés à des arbres de défaillance pour le parcours des événements, il établit la localisation de la défaillance.

Dans [GEN 03], un RdP labellisé modélise le procédé en décrivant son comportement normal et anormal. Un label, ou étiquette, est une indication sur le type d'événement présent sur les transitions du RdP qu'il soit observable ou non observable. De ce RdP, [GEN 03] exprime un diagnostiqueur décrivant toutes les situations observables possibles à partir d'une situation afin de pouvoir identifier tous les types de défauts (Figure 2-5).

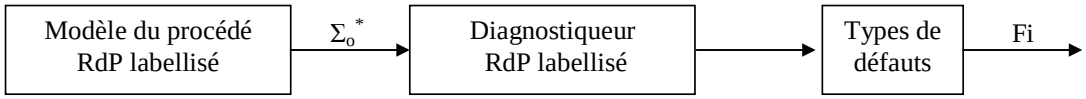
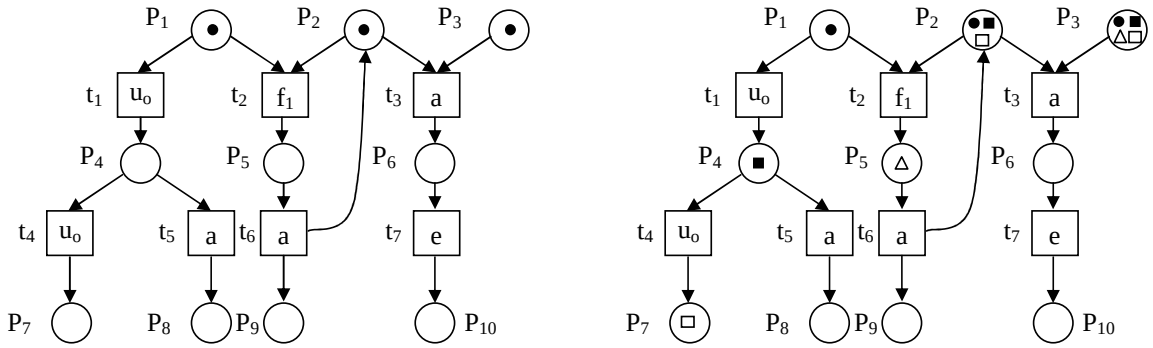


Figure 2-5 : Approche par RdP labellisé

Prenons l'exemple de la Figure 2-6a représentant un RdP modélisant le fonctionnement normal et anormal d'un procédé.



a) Modèle du procédé

b) Diagnostiqueur

$$X_{d0} = \begin{array}{c} \begin{array}{cccccccccc|cc} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & f_1 & f_2 \end{array} \\ \left(\begin{array}{cccccccccc|cc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array} \begin{array}{c} \bullet \\ \blacksquare \\ \Delta \\ \square \end{array}$$

c) Matrice d'états du diagnostiqueur

Figure 2-6 : Diagnostic par Réseaux de Petri

A partir des places P_1 et P_2, il est possible soit de tirer la transition t_2 possédant un label, ou étiquette, indiquant que le défaut f_1 a eu lieu et aboutissant à la place P_5, soit de tirer t_1 correspondant à un label d'événement non observable u_o pour arriver à la place P_4. Pour isoler ce défaut, il faut alors pouvoir connaître toutes les situations du système. A partir du modèle du procédé, [GEN 03] réalise directement un diagnostiqueur sous la même forme que le RdP du procédé (Figure 2-6b). Cependant, de ce diagnostiqueur, il établit les différentes configurations du procédé à partir des observations possibles par différents types de jetons (Δ,•,■,□). Ces jetons expriment tous les états initiaux du procédé. Ainsi, à partir du marquage initial du modèle RdP du procédé, il est possible d'avoir à l'initialisation :

- P_1, P_2, P_3 par le jeton de type • si aucun événement n'a encore eu lieu
- P_4, P_2, P_3 par le jeton de type ■ si l'événement non observable u_o a eu lieu
- P_5, P_3 par le jeton de type Δ si l'événement de défaut f_1 a eu lieu
- P_7, P_2, P_3 par le jeton de type □ si l'événement u_o a eu lieu deux fois.

A partir de là, il est possible de définir une matrice des états initiaux X_{d0} représentant toutes les situations initiales possibles avec l'affectation des défauts qui leur est associée (Figure 2-6c). Par exemple, la situation initiale représentée par le symbole "•" correspond à la première ligne de la matrice où le procédé se trouve dans les places P_1, P_2 et P_3 et où aucun défaut n'est apparu. Le symbole "Δ" décrit, quant à lui, une situation initial du procédé où le défaut f_1 est apparu puisque la matrice d'états du diagnostiqueur indique un "1" logique dans le colonne f1 en regard de la situation P_3 et P_5 du modèle du diagnostiqueur. Ainsi, pour chaque évolution observable du procédé, il faut établir le diagnostiqueur et la matrice d'états correspondante qui permet d'identifier les défauts pour chaque situation possible.

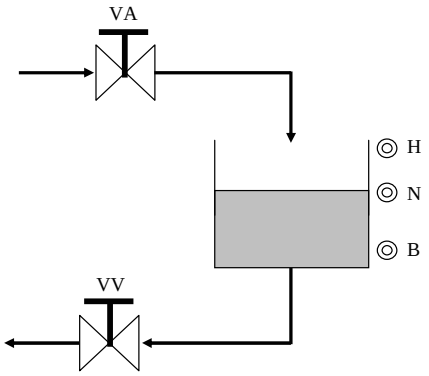
L'avantage des approches à base de RdPs est lié aux outils mathématiques supportés par le modèle. Ainsi, les matrices d'incidences avant et arrière permettent de réaliser des estimations de séquences. De plus, les RdPs sont des modèles très évolués gérant les procédés concurrentiels nécessitant d'établir une distribution des éléments.

2.2.1.3. Diagnostic à base d'expressions logiques

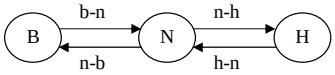
La modélisation d'un SED par un automate permet une interprétation de la situation d'un système mais est souvent difficilement exploitable dans le cas des systèmes complexes où le nombre d'états est important. La modélisation par un automate peut être renforcée en exploitant les propriétés algébriques du procédé. En effet, dans le cas où le procédé est composé de plusieurs capteurs et actionneurs de type Tout Ou Rien, renvoyant une valeur 0 ou 1, la modélisation par algèbre de Boole est particulièrement intéressante. Cet outil de représentation décrit le comportement du système par des spécifications en utilisant des prédicats booléens. C'est un modèle abstrait permettant une exploitation et une compréhension souvent plus aisées. Chaque état du procédé est représenté par un vecteur composé de variables booléennes. Le passage d'un état à un autre ne se produit que par changement d'une variable.

[WAN 00] présente, dans ses travaux, une approche booléenne permettant de représenter la situation d'un système autour de plusieurs exemples dont notamment un système de régulation de niveau d'une cuve (Figure 2-7a) par le vecteur de ses observations qui évolue en fonction des ordres émis. Ces travaux s'inscrivent plus dans le cadre de la synthèse de la commande afin de construire un superviseur. Cependant, les expressions logiques peuvent être utilisées dans le cadre de la détection de défauts en utilisant l'estimation des états du comportement normal.

L'automate de la Figure 2-7b modélise le passage d'un niveau de la cuve à un autre. Ainsi, l'état décrivant le niveau haut de la cuve H peut évoluer vers le niveau normal N par l'événement h-n représentant le passage du niveau haut à normal. Il en est de même pour le passage du niveau normal N vers le niveau bas B par l'événement n-b. On comprend alors que le passage du niveau bas B vers le niveau normal N est effectué sur l'événement b-n et que le passage du niveau normal N vers le niveau haut H est effectué sur l'événement n-h.



a) Système de remplissage d'une cuve



b) Automate d'évolution du niveau de la cuve

Vecteur d'état X	↑VA	↓VA	↑VV	↓VV	b-n	n-b	n-h	h-n
H	0	0	0	0	0	0	1	1
N	0	0	0	0	1	1	1	1
B	0	0	0	0	1	1	0	0
VA	1	1	0	0	0	0	0	0
VV	0	0	1	1	0	0	0	0

c) Table d'évolution de la situation courante X

VA : Vanne d'Alimentation
VV : Vanne de Vidange
H : Niveau Haut de la cuve
N : Niveau Normal de la cuve
B : Niveau Bas de la cuve
b-n : Passage du niveau Bas vers le niveau Normal
n-b : Passage du niveau Normal vers le niveau Bas
n-h : Passage du niveau Normal vers le niveau Haut
h-n : Passage du niveau Haut vers le niveau Normal

d) Dénomination des symboles de la cuve

Figure 2-7 : Procédé et évolution du niveau d'une cuve

La table d'évolution de la situation courante X décrit les interactions de cause à effet entre les événements et les états (Figure 2-7c). Le symbole ↑ indique le passage d'un capteur, ou d'un ordre, de son état inactif 0 à l'état actif 1, alors que le symbole ↓ indique le contraire. Ainsi, l'événement b-n, indiquant le passage du niveau bas B vers le niveau normal N, influe sur l'état représentant le niveau bas B et le niveau normal N. Cette interaction est représentée par un "1" logique dans la table. Il est possible alors de constituer un vecteur d'évolution E = (H, N, B, VA, VV) indiquant la variable sur laquelle un événement survient par le changement de sa valeur. Par exemple, le vecteur d'évolution pour l'événement ↑VA est donné par la première colonne de la table d'évolution et correspond à $E_{\uparrow VA}$ = (0 0 0 1 0). Ce vecteur indique bien qu'un événement est survenu et que la variable VA a changé de valeur.

L'ensemble des états courants du procédé est représenté par un vecteur d'état X = (H, N, B, VA, VV). Ce vecteur permet d'obtenir la description complète du procédé. L'évolution future Y du procédé à partir du vecteur d'état courant X est obtenue par estimation de l'état à l'aide de l'opérateur "OU-exclusif" par l'équation $Y = X \oplus E$. En effet, si l'état actuel est X = (0 0 1 1 0) et que l'événement qui survient est la fermeture de la vanne d'alimentation équivalent à l'événement ↓VA, alors le vecteur d'évolution $E_{\downarrow VA}$ est donc (0 0 0 1 0) indiquant bien un changement de la valeur de la variable VA. Il est alors possible de décrire l'état futur Y par $Y = X \oplus E_{\downarrow VA}$ = (0 0 1 1 0) ⊕ (0 0 0 1 0) soit Y = (0 0 1 0 0).

Cette modélisation par expression logique est efficace dans le cadre de la détection de défauts car elle détecte directement le changement de chaque événement observable. Elle permet également de réaliser une estimation du comportement du procédé sans réaliser de modèles sous forme d'automates. Cependant, les expressions booléennes seules perdent l'aspect séquentiel des procédés. C'est pourquoi, il est préférable d'allier les automates aux expressions booléennes pour obtenir un automate booléen équivalent.

Pour l'exemple du niveau de la cuve de la Figure 2-7a, l'automate booléen est présenté en Figure 2-8 à travers 12 états où le vecteur d'état évolue en fonction des événements qui occurrent.

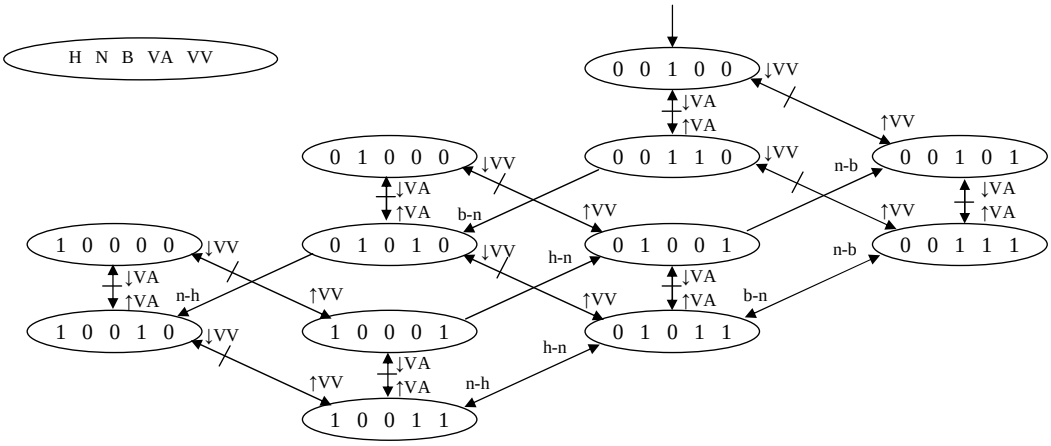


Figure 2-8 : Modélisation booléenne sous forme d'automate

2.2.1.4. Diagnostic à base de chroniques ou de templates

Les automates et RdPs modélisent essentiellement les procédés à travers des événements et leur ordre d'occurrence. Cependant, il est souvent nécessaire de connaître le temps de travail sur un produit et/ou d'établir des contraintes temporelles sur les différentes activités. Il faut alors faire appel à des modèles temporels. Un modèle temporel représente des relations temporelles exprimant des contraintes temporelles entre les événements du système à diagnostiquer. Ces contraintes définissent un ensemble de relations entre les dates d'occurrence d'événements. Un modèle temporel peut être modélisé par :

- Un automate temporel (délai constant) [RIC 01],
- un automate temporisé (délai borné) [PAN 00],
- des treillis temporels (contraintes sur intervalles) [GHA 96],
- un RdP temporel [BOU 03a], [BOU 03b].

Une contrainte temporelle peut être associée soit au matériel, capteurs et actionneurs, soit au produit. Il faut définir l'objectif du diagnostic, à savoir si le produit intervient ou non dans la modélisation du procédé. On parle alors de procédés "à *instance-unique*" et de procédés dit "*à instance-multiple*".

Les procédés à *instance-unique* possèdent une séquence d'événements se répétant toujours de la même façon avec certaines restrictions temporelles impliquant un délai entre l'occurrence de deux événements. Ils sont représentatifs des systèmes manufacturiers ayant des composants physiques à modéliser. Les procédés à *instance-multiple* sont les procédés devant réaliser la production de plusieurs produits générant des séquences d'événements fixes pour chaque produit mais modulables sur l'ensembles des observations. Ils sont donc représentatifs des systèmes manufacturiers gérant la production de produits.

La Figure 2-9 est un exemple de système manufacturier tiré de [PAN 00] composé d'un vérin devant transférer des caisses d'un tapis A vers un tapis B. Prenons le vérin pour illustrer l'aspect *instance-unique* du procédé. Le vérin est commandé en sortie par l'activation d'un événement *S* qui engendre l'activation de son capteur de fin de course de sortie par l'événement *b*. La rentrée du vérin est, quant à elle, obtenue par l'activation de l'événement *R* qui entraîne l'activation de son capteur de fin de course de rentrée par l'événement *a*. Le

fonctionnement normal du système implique une séquence d'événements *SbRa* répétée sur le vérin. L'observation d'une séquence d'événements autre que celle attendue, représente l'apparition d'un défaut. Cette séquence doit être satisfaite dans un temps donné par l'ajout de contraintes temporelles sur les événements. Par exemple, il existe un délai qui doit être respecté entre l'activation de l'ordre de sortie *S* et l'activation de son capteur de fin de course de sortie *b*.

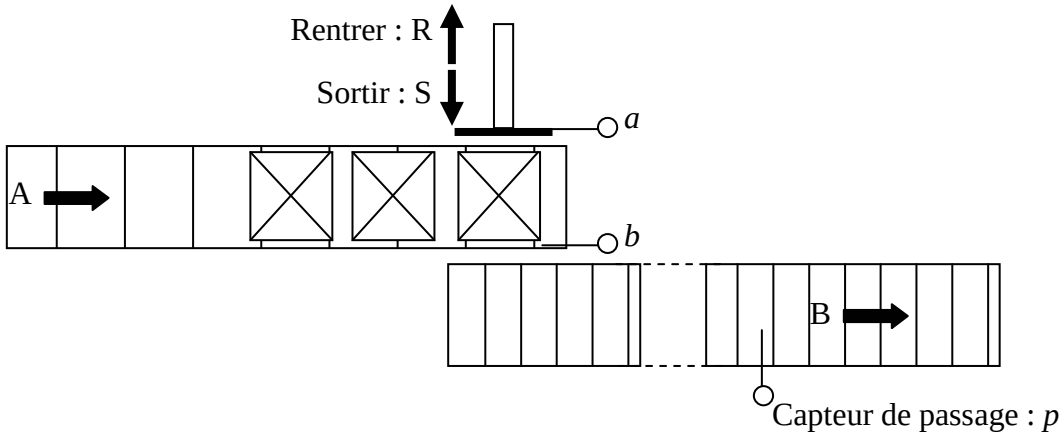


Figure 2-9 : Exemple de système manufacturier

Pour illustrer maintenant l'aspect *instance-multiple* d'un procédé, considérons la partie du tapis B de l'exemple de la Figure 2-9 de [PAN 00]. On supposera que lorsqu'une caisse présente sur le tapis A est poussée sur le tapis B par le vérin (sortie de tige détectée par *b*), la caisse mette entre 15 et 17 secondes pour arriver jusqu'au capteur de passage *p*. Chaque caisse va alors générer une séquence d'événements *bp* de 15 à 17 secondes entre les événements *b* et *p*.

Cependant, plusieurs caisses sont en traitement permanent, il est donc par conséquent possible d'avoir un nombre indéterminé de caisses sur le tapis B entre *b* et *p*. L'observation du procédé ne permet pas alors de vérifier directement la séquence *bp* mais peut aboutir à l'observation entrelacée de paires d'événements *b* et *p* et donner par l'occasion une séquence d'événements *bpbbpbpp* tout en respectant les délais entre les événements comme montré en Figure 2-10.

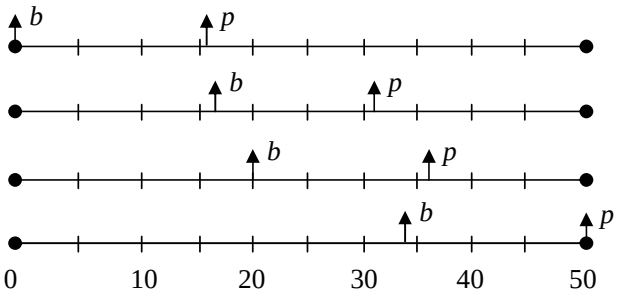


Figure 2-10 : Illustration d'un procédé à instance-multiple

Les contraintes de temps doivent être appliquées sur les séquences d'événements locales au produit et non pas sur l'observation des séquences globales du procédé. Le comportement d'un procédé à *instance-multiple* sur le produit ne peut donc pas être modélisé comme un procédé à *instance-unique* sur le matériel.

L'intégration du temps dans un procédé à *instance-unique* est réalisable par des modèles temporels. Par contre, l'application de contraintes temporelles sur un procédé à *instance-multiple* est très délicate puisqu'il faut réaliser non seulement un modèle de PO, mais également un modèle du produit. Des méthodes utilisant des contraintes temporelles pour les procédés à *instance-multiple* sont utilisées dans la littérature. Nous présentons, dans ce paragraphe, deux d'entres elles qui sont associées à des modèles SED : les chroniques et les "*templates*" (prévisions).

2.2.1.4.1. Diagnostic à base de chroniques

Une chronique est un graphe définissant des contraintes temporelles entre les dates d'occurrence d'un ensemble d'événements généré pour un procédé [GHA 96]. Une chronique est associée à un produit dont l'information temporelle est recueillie en fonction de sa production. L'aspect temporel est pris en compte à travers la spécification de contraintes temporelles entre les différents événements du procédé. Dès lors, les différentes chroniques doivent être cohérentes entre elles et la fonction de détection a alors pour rôle de reconnaître les évolutions défaillantes. Ces chroniques peuvent être modélisées par un RdP temporel comme dans le cas des travaux de [BOU 03a], [BOU 03b] où le franchissement d'une transition représente l'occurrence d'un événement associé à une contrainte temporelle. L'utilisation d'une chronique par un RdP permet la gestion des procédés à *instance-multiple* puisqu'il est possible d'avoir plusieurs jetons pour une même place, ce qui n'est pas possible pour les automates à états. La Figure 2-11 présente une modélisation d'une chronique où, à partir de la place P_1, il est possible d'avoir l'événement e_1 dans l'intervalle de temps [3, 8] pour arriver à la place P_2. Un jeton, dans cette chronique, peut ainsi représenter un produit devant respecter chaque contrainte temporelle. Plusieurs produits de types différents peuvent être représentés par des jetons de couleurs différentes à travers un RdP temporel coloré.

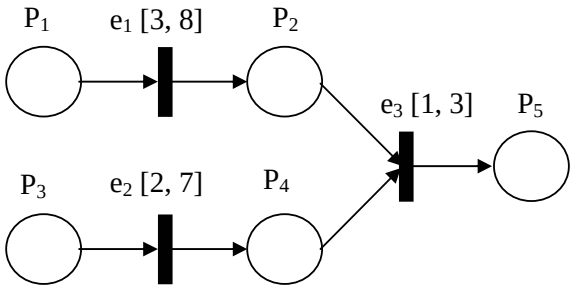


Figure 2-11 : Modélisation d'une chronique

2.2.1.4.2. Diagnostic à base de *templates*

[HOL 94] et [PAN 00] utilisent un automate temporisé pour représenter les relations temporelles entre les événements et le séquencement entre ces événements pour les deux cas (à *instance-unique* et à *instance-multiple*). Des *templates*, basées sur ces automates, sont construites afin de définir pour chaque événement observé les conséquences normales futures et les conséquences anormales correspondant à un défaut. Une *template* est de la forme (t, *e*, C, w) où t est l'instant d'occurrence de l'événement $e \in \Sigma$, C est l'ensemble des conséquences, et w est une étiquette. Une conséquence est une paire (e', τ) où $e' \in \Sigma$ et τ est l'intervalle de délai à borne positive ou négative. Une *template* (t, *e*, C, w) est satisfaite sous une trace ρ et une période $]0, t_f]$ s'il existe une conséquence $(e', \tau') \in C$ et $t' \in [t] \oplus \tau'$ tel que la conséquence (e', t') doit avoir lieu dans un intervalle $]0, t_f]$, où t_f est le retard maximum et $\oplus$ est la somme des intervalles.

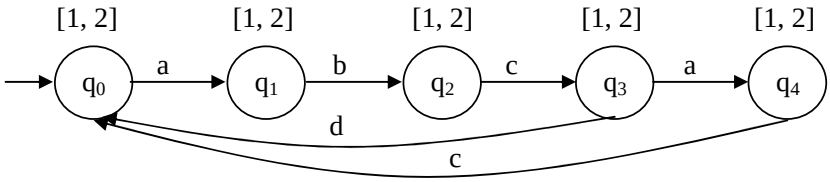


Figure 2-12 : Exemple d'automate temporisé

L'automate temporisé seul ne peut pas gérer les cas des procédés à *instance-multiple* puisqu'il est impossible d'évoluer sur plusieurs états dans un même automate [PAN 00], à l'inverse du RdP. Les *templates* permettent le diagnostic des procédés concurrentiels de production de plusieurs types de produits puisqu'un produit est associé à un ensemble de *templates*. Associé à un automate à état temporisé, il est alors possible de réaliser un diagnostic multi-produits avec contraintes temporelles.

Prenons l'exemple de la Figure 2-12 représentant un modèle du procédé à *instance-unique*. L'automate est caractérisé par une contrainte de temps dans chaque état. Dès lors, chaque événement va être associé à des *templates* précisant l'état d'où il vient, les états dans lesquels il peut se rendre et les contraintes de temps à sa transition de sortie (Tableau 2-2). Un produit va comporter une série de *templates* positives à satisfaire (1, 2, 3, 4) du Tableau 2-2 et une autre série de *templates* négatives à ne pas satisfaire (5, 6, 7, 8) du Tableau 2-2. Les *templates* positives sont les conséquences postérieures à l'état. Ce sont les événements devant avoir lieu dans l'intervalle $]0, t_f]$. Une conséquence postérieure est définie par $C_p(q) = \{(e, d(q)) \mid \forall e$ sortant de $q\}$ où $d(q)$ est l'intervalle associé à l'état q. L'ensemble des conséquences postérieures est l'ensemble des événements de sortie de l'état q et des durées pendant lesquelles ils peuvent être générés par le procédé. Les *templates* négatives sont les conséquences interdites. Elles permettent de détecter si un événement a eu lieu plus tôt qu'il ne fallait. Une conséquence interdite est définie par $C_i(q) = \{(e,]-\min(d(q)), 0[) \mid \forall e$ entrant à $q\}$. L'ensemble des conséquences interdites est l'ensemble des événements d'entrée de l'état q associé aux durées séparant ces événements aux événements de sortie et pendant lesquelles ils ne doivent pas être générés.

1	(a, [{q_0}, {b, [1, 2]}, q_1], [{q_3}, {c, [1, 2]}, q_4])
2	(b, [{q_1}, {c, [1, 2]}, q_2])
3	(c, [{q_2}, {(a, [1, 2]), (d, [1, 2])}, q_3], [{q_4}, {a, [1, 2]}, q_0)])
4	(d, [{q_3}, {a, [1, 2]}, q_0])
5	(a, [{q_0}, {(c,]-1, 0[), (d,]-1, 0[)}, *pb*], [{q_3}, {c,]-1, 0[}, *pb*])
6	(b, [{q_1}, {a,]-1, 0[}, *pb*])
7	(c, [{q_2}, {b,]-1, 0[}, *pb*], [{q_4}, {a,]-1, 0[}, *pb*])
8	(d, [{q_3}, {c,]-1, 0[}, *pb*])

Tableau 2-2 : Exemple de templates associées à l'automate temporisé

Prenons par exemple de la Figure 2-12, les templates positives vont être de type (e, [{q}, {e', $d(q)$}, q']). La *template* 1 exprime les conséquences postérieures où l'événement *a* peut provenir soit de l'état q_0 et arrivant à l'état q_1 par l'événement *b* dans un intervalle de temps [1, 2], soit de l'état q_3 conduisant à l'état q_4 par l'événement *c* qui doit arriver dans l'intervalle

de temps [1, 2]. Les *templates* négatives (*e*, [{*q*}, {(*e'*, -min]d(*q*)), 0[}, *pb*]) expriment par l'étiquette *pb* un problème où *e* est l'événement déclencheur. Ainsi, la *template* 6 manifeste le fait que l'événement *b* ne doit pas survenir si l'événement *a* apparaît dans un retard minimal compris dans]-1, 0[pour arriver à l'état q_1. Si la *template* est satisfaite, alors un défaut est détecté.

L'avantage des *templates* réside dans le fait qu'elles peuvent être associées à une modélisation du produit. En effet, la plupart des outils de modélisation pour le diagnostic cherche à représenter le comportement du procédé à travers leurs états. Les *templates* peuvent, par contre, décrire le comportement du produit traité. Chaque produit doit respecter les *templates* malgré l'aspect concurrentiel des événements. C'est donc dans une optique de diagnostic des systèmes manufacturiers à *instance-multiple* qu'elles sont notamment efficaces.

2.2.2. Modélisation des défauts et notations

La plupart des modèles cherche à décrire soit le comportement de la PO, soit le comportement désiré de la PC. Un défaut est vu comme l'occurrence d'un événement non observable. Sa détection est difficile et suppose une modélisation bien plus complexe que celle de la PO et de la PC. En effet, un défaut ne peut être détecté qu'après un nouvel événement observable du procédé. La détection fait appel à des modèles de défauts qui doivent interpréter les séquences d'événements ou à un état du procédé à travers son fonctionnement normal et défaillant. On parle alors de diagnostiqueur. La modélisation des défauts suppose par conséquent une connaissance a priori des défauts que l'on souhaite détecter et diagnostiquer. Ces défauts sont répartis dans plusieurs partitions, $\Sigma_{\Pi} = \{\Pi_{F1}, \Pi_{F2}, \ldots, \Pi_{Fr}\}$. Chaque partition Π_{Fi} regroupe tous les défauts qui soit ont le même effet sur le procédé, soit la procédure de reprise à faire, après l'occurrence d'un de ces défauts, est la même.

A chaque partition de défauts Π_{Fi} correspond une étiquette Fi appartenant à l'ensemble des étiquettes de comportement défaillant Λ_F = {F1, F2, …, Fr}. L'ajout de l'étiquette N à Λ_F, indiquant un fonctionnement normal du procédé, conduit à l'obtention de l'ensemble des étiquettes Λ de tout comportement possible du procédé. Le comportement défaillant peut s'exprimer soit par l'apparition d'un événement de défaut simple et retourne une étiquette Fi, soit par l'apparition d'un défaut multiple correspondant à la propagation de plusieurs défauts et retournant plusieurs étiquettes différentes Λ_F.

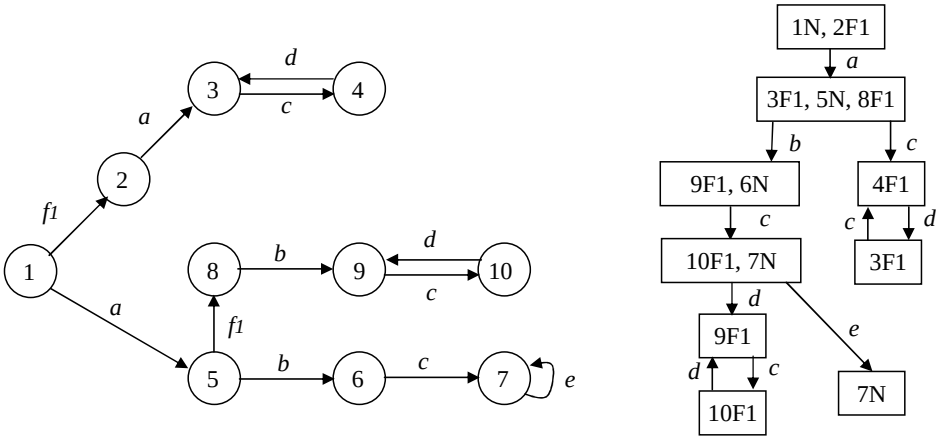
Un défaut est modélisé soit comme la conséquence de l'exécution d'un événement, modélisation à base d'événements, soit l'atteignabilité à un état défaillant, modélisation à base d'états, ou encore en combinant les deux modélisations (mixte). Chacune de ces modélisations conduit à l'obtention d'un diagnostiqueur à base d'événements, à base d'états ou mixte.

2.2.2.1. Modélisation à base d'événements

Une séquence de défaillance est considérée comme une séquence d'événements comprenant au moins un défaut. La modélisation à base d'événements considère que la détection d'un défaut est réalisée par la détection d'une séquence de défaillance. L'analyse de cette séquence permet de détecter l'apparition d'un défaut lorsqu'un nouvel événement

observable caractérisant ce défaut survient. A partir d'un modèle de procédé représentant le comportement normal et défaillant, il faut pouvoir créer un modèle permettant d'étudier toutes les séquences observables du procédé et de déterminer si une de ces séquences est normale, défaillante ou correspond à un cas incertain signifiant une indécision : c'est le rôle du diagnostiqueur [SAM 95]. Un état incertain du diagnostiqueur est un état possédant plusieurs étiquettes différentes.

La Figure 2-13a présente la modélisation à base d'événements d'un procédé à travers un automate à états pour la détection d'un événement f_1 appartenant à la partition de défaut Π_{F1}. Dès lors, ce modèle dispose seulement d'informations provenant des événements. Ces états sont, par conséquent, difficilement interprétables. En effet, à partir de l'état initial, il est impossible de savoir si le système se trouve dans l'état 1 en fonctionnement normal ou dans l'état 2 indiquant l'occurrence du défaut f_1. De même, à partir de l'occurrence de l'événement *a*, il est difficile de savoir si le procédé se trouve dans l'état 5 indiquant un fonctionnement normal du procédé ou dans l'état 8 indiquant un défaut de type F1.



a) Modèle du procédé b) Diagnostiqueur

Figure 2-13 : Modélisation à base d'événements

Il faut faire appel à un diagnostiqueur ayant la connaissance de l'accessibilité des états. Chaque état x du diagnostiqueur à une fonction de décision $l(x) \in \Delta$, où Δ est l'ensemble de tous les sous-ensembles des étiquettes Δ = {{N}, {F1}, {F2}, …, {Fr}, {N, F1}, {N, F2}, …, {N, Fr}, …, {N, F1, F2}, …, {N, F1, F2, …, Fr}, {F1, F2}, {F1, F2, F3}, …, {F1, F2, …,Fr}}. On peut donc dire que $|\Delta| = 2^{|\Lambda|} - 1$. Tout état x ayant l(x) qui possède une seule étiquette de défaut est un état certain. Un diagnostiqueur $Gd = (X_d, \Sigma, \delta_d, x_{d0})$ est un automate réduit aux événements observables avec mémoire des événements de défauts représentés par des étiquettes indiquant le fonctionnement normal ou défaillant.

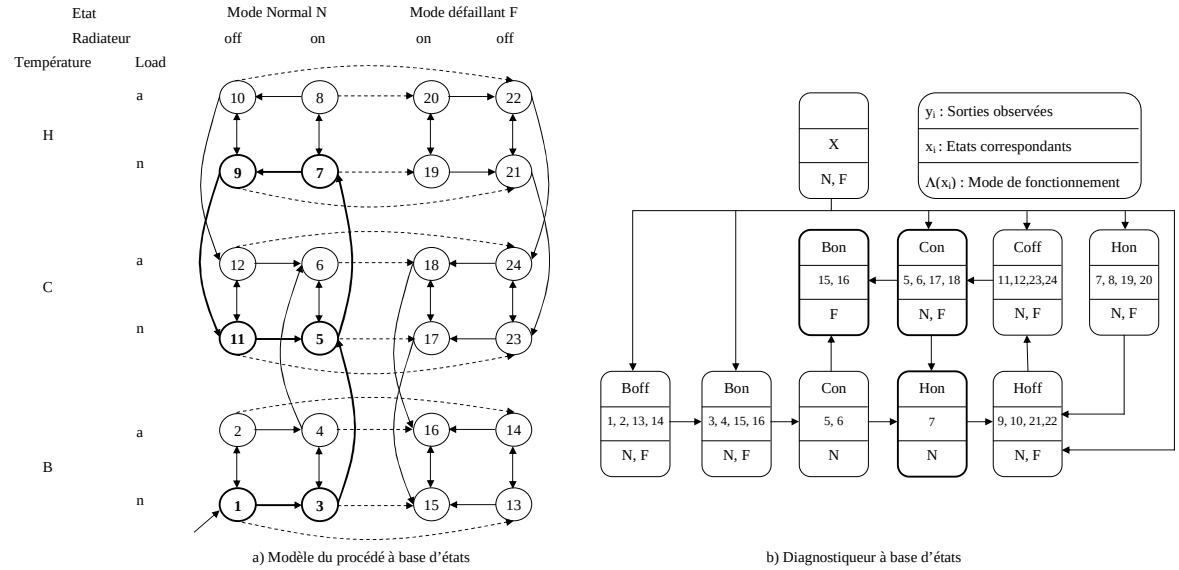
Pour le modèle du procédé de la Figure 2-13a, le diagnostiqueur obtenu est présenté sur la Figure 2-13b. Ainsi, à partir de l'état 1 du modèle du procédé, l'état 1N du diagnostiqueur est construit. Suite à l'occurrence de l'événement *a*, le procédé arrive soit à l'état 3, soit à l'état 5. Le diagnostiqueur indique alors que cet événement permet d'accéder soit à l'état 3 en détectant un événement de défaut f_1 par le label, ou l'étiquette, F1, soit à l'état 5 en fonctionnement normal par l'étiquette N. Cet état du diagnostiqueur ne permet cependant pas d'isoler le défaut et de s'assurer du bon fonctionnement du procédé. C'est après l'occurrence de l'événement *c* que l'événement de défaut f_1 est détecté par l'étiquette F1. C'est l'occurrence de l'événement *c* qui permet d'isoler le défaut. Cependant, si l'événement *b*

arrive à partir de l'état incertain {3F1 5N} du diagnostiqueur, alors on constate à nouveau un état incertain {9F1 6N}. Dès lors, il faut attendre l'occurrence des événements *c* puis *d* pour isoler le défaut dans un état certain, x = 9, du diagnostiqueur pour lequel l = {F1}.

Cependant, le diagnostiqueur doit être initialisé en même temps que le procédé afin qu'il puisse suivre l'évolution du procédé. Cette initialisation est difficile à obtenir pour les systèmes manufacturiers. En effet, lorsqu'un défaut survient et demande un arrêt complet du système, la procédure de remise en route consiste très souvent à réinitialiser la commande, la partie opérative et également les diagnostiqueurs. Cependant, il n'est pas possible de connaître la situation de ces derniers puisqu'ils sont modélisés à travers leurs événements et non pas leurs états.

2.2.2.2. Modélisation à base d'états

La modélisation à base d'états considère que l'occurrence d'un défaut est détectée par l'arrivée dans un état caractérisant ce défaut. [ZAD 03] a défini un diagnostiqueur à base d'états caractérisé par les sorties des capteurs et des commandes ainsi que le chemin qui conduit à ces états normaux ou défaillants. Il explique cette modélisation autour d'un système de régulation de température d'une salle composée d'un radiateur, d'un capteur de température et de la commande du radiateur. Le modèle du procédé global est représenté en Figure 2-14a où les arcs en pointillés représentent un événement de défaut du radiateur. Le label « Load » modélise les effets de la perturbation sur le capteur de température dus à la température ambiante ou aux transferts de chaleur des salles voisines. Les perturbations sont alors représentées par deux états : *n* pour normal et *a* pour perturbé. Elles sont considérées comme des perturbations compensables par le régulateur. L'hypothèse faite est de ne pas les prendre en compte et qu'en présence de perturbation, la température garde sa consigne.



a) Modèle du procédé à base d'états

b) Diagnostiqueur à base d'états

Figure 2-14 : Modélisation à base d'états

Les six situations du procédé correspondent aux sorties observables suivantes :

- Boff : température basse (B), radiateur éteint (off)
- Bon : température basse (B), radiateur allumé (on)

- Coff : température sous la consigne (C), radiateur éteint (off)
- Con : température sous la consigne (C), radiateur allumé (on)
- Hoff : température au dessus de la consigne (H), radiateur éteint (off)
- Hon : température au dessus de la consigne (H), radiateur allumé (on)

Le modèle du procédé est alors composé de l'ensemble des états X = {1, 2, …, 24}, de l'ensemble des sorties observables des états Y = {Boff, Bon, Coff, Con, Hoff, Hon} et de l'ensemble des labels, ou étiquettes, de fonctionnement normal et défaillant Λ = {N, F} où $l(x_i)$ = N pour les états de $1 \leq i \leq 12$ et $l(x_i)$ = F pour les états de $13 \leq i \leq 24$. Ainsi, à partir de l'état 1 correspondant à la situation du radiateur éteint avec une température basse sans perturbation, il est possible d'évoluer vers l'état 3 où le radiateur est allumé. La température se trouve sous la consigne dans l'état 5. Lorsque la température dépasse la consigne alors le procédé est dans l'état 7 puis dans l'état 9 afin d'arrêter le radiateur et de redescendre en dessous de la consigne en état 11 (états en gras sur la Figure 2-14a). La régulation de la température est alors représentée par le retour à l'état 5. Le modèle indique bien qu'à partir de chaque état, il est possible d'avoir un défaut et de retrouver le procédé dans un mode défaillant F qui correspond à une défaillance de la résistance de chauffe du radiateur.

A partir du modèle du procédé à base d'états, [ZAD 03] construit un diagnostiqueur à base d'états (Figure 2-14b). Le diagnostiqueur est composé d'états enrichis indiquant les sorties observées y_i, les états correspondant x_i et le label du mode de fonctionnement du procédé $l(x_i) \in \Lambda$. Ainsi, pour la sortie Con, les états accessibles sont {5, 6, 17, 18} et le mode est soit normal N ou défaillant F. A partir de ces états, il est possible d'évoluer soit vers l'état 7 représenté par la sortie Hon, soit vers les états 15 et 16 associés à la sortie Bon (états en gras sur la Figure 2-14b). Dès lors, les états 15 et 16 représentant un défaut, l(15) = l(16) = F, sont isolés de l'état 7 représentant un fonctionnement normal, l(7) = N. Cet exemple, proposé dans [ZAD 03], ne traite pas de l'ensemble des défauts possibles et notamment des défauts intermittents. Il suppose d'avoir des capteurs robustes qui ne sont pas considérés comme susceptibles d'être défaillants.

Ce type de modélisation a l'avantage de ne pas avoir besoin d'initialiser le diagnostiqueur en même temps que le procédé. L'inconvénient de la modélisation à base d'états est qu'elle ne peut détecter les pannes intermittentes qui sont des événements brefs amenant à des états non stables, donc non représentatifs.

2.2.2.3. Modélisation mixte

La modélisation à base d'événements permet de détecter tout type de défaut, capteur et/ou actionneur, même dans le cas de panne intermittente. Cependant, il faut attendre une observation alors que le procédé se trouve peut être déjà dans une situation anormale. Cette contrainte implique qu'il faut pouvoir initialiser le procédé et le diagnostiqueur au même instant et de connaître cet état initial. A contrario, la modélisation à base d'états regarde l'atteignabilité des différents états de défaut sans se préoccuper des différents événements amenant à ce dysfonctionnement. Dès lors, le procédé peut se trouver dans n'importe quel état dès le démarrage du diagnostiqueur. La modélisation mixte consiste par conséquent à établir un lien entre l'occurrence des événements et l'atteignabilité des états de défaut. Cette modélisation permet ainsi de diagnostiquer un maximum de défauts en paliant aux problèmes de la modélisation à base d'événements et à base d'états.

Une approche de modélisation mixte sera développée dans le chapitre 3 afin de pouvoir à la fois détecter les pannes permanentes et intermittentes à l'aide de la modélisation à base d'événements, tandis que les états permettront de ne pas se préoccuper de l'initialisation du ou des diagnostiqueur(s) (Tableau 2-3). Cette modélisation mixte est intéressante pour le cas des systèmes manufacturiers puisque leur état initial peut être inconnu.

	A base d'événements	A base d'états	Mixte
Pannes permanentes	OK	OK	OK
Pannes intermittentes	OK	Problème	OK
Initialisation	Problème	OK	OK

Tableau 2-3 : Comparatif des modélisations à base d'événements, d'états et mixte

2.2.3. Structure de la prise de décision

Dans le chapitre 1, les structures pour le diagnostic des défaillances ont été présentées brièvement. Dans cette partie, une comparaison détaillée autour d'un exemple reprend ces structures afin d'en présenter les avantages et les inconvénients.

2.2.3.1. Structure centralisée

La structure centralisée d'un diagnostiqueur consiste à représenter un seul modèle du procédé avec un seul diagnostiqueur. Reprenons l'approche de [SAM 95] évoquée au paragraphe 2.4.1.1 pour illustrer cette structure. Cette approche souhaite établir le diagnostic d'un procédé à N composants. La première étape consiste à modéliser chaque composant par un automate à état, défini par $G_i = (X_i, \Sigma_i, \delta_i, x_{0i})$ permettant de modéliser le $i^{ième}$ composant. Les états X_i et les événements Σ_i reflètent le comportement normal et anormal du $i^{ième}$ composant. Les événements de défaillance sont inclus dans les événements non observables de Σ_i. Pour obtenir le modèle global du système, il suffit alors de réaliser une composition synchrone des différents automates à état de chaque composant avec celui de la commande. Le résultat de cette opération est alors un automate $G = (X, \Sigma, \delta, x_0)$. A partir de cet automate, il faut pouvoir établir un diagnostiqueur à base d'événements $G_d = (X_d, \Sigma, \delta_d, x_{d0})$ permettant de détecter l'ensemble des partitions de défauts Σ_{fi} du modèle G. Ce diagnostiqueur décrit alors chaque situation accessible du modèle global à partir des événements observables en précisant à chaque fois à travers une fonction de décision, $l(x_i) \in \Delta$, les situations normales et défaillantes.

Reprenons l'exemple de la Figure 2-3 modélisant trois composants d'un système de distribution d'eau avec sa commande. Le modèle global est obtenu par composition synchrone des différents composants en Figure 2-4. Pour obtenir le diagnostiqueur global, il faut définir une partition de défaut $\Pi_{F1} = \{f_1, f_2\}$ avec l'événement de défaut $f_1 = \{FC\}$ représentant un défaut lorsque la vanne est fermée et collée, et l'événement de défaut $f_2 = \{OC\}$ pour un défaut de vanne ouverte et collée. La Figure 2-15 représente alors le diagnostiqueur *Gd* de ce système de distribution d'eau. A partir de l'état 1 du modèle du procédé, l'état du diagnostiqueur 1N est construit, il est possible d'avoir après l'observation de l'événement *conso*, la situation {3N, 5F1} qui indique que le modèle du procédé est soit dans l'état 3 en

situation normale, soit dans l'état 5 avec un défaut de type F1. Après chaque événement observable, le diagnostiqueur observe les états atteignables du procédé. De l'état {3N, 5F1}, le procédé va dans l'état {7N, 9F1} après le démarrage de la pompe par l'événement DP. L'ouverture de la vanne est accompagnée d'une indication de présence de flux par le capteur de flux, alors cela signifie que le procédé est en situation normale {11N}. Dans le cas où aucun flux n'est détecté, alors la demande d'ouverture de vanne OV est l'observation permettant d'isoler le défaut F1 par l'état {13F1}. L'ensemble des états du diagnostiqueur est construit de la même manière.

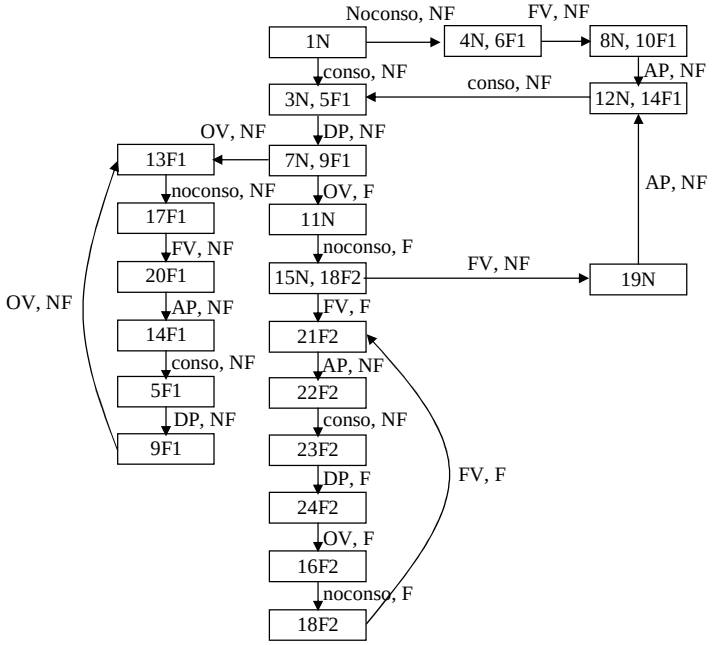


Figure 2-15 : Diagnostiqueur G_d pour le système de distribution d'eau de la Figure 2-3

Cette approche de diagnostiqueur autour d'une structure centralisée est cependant soumise au problème de l'explosion combinatoire. De plus, pour les systèmes manufacturiers, les éléments ont une nature informationnelle décentralisée. C'est pourquoi de nombreux travaux ont abandonné cette approche centralisée au profit d'approches décentralisées, voire distribuées.

2.2.3.2. Structure décentralisée sans coordinateur

La structure décentralisée consiste à partir du modèle global du procédé à obtenir des observations locales par projection naturelle (voir Annexe 1) pour établir des diagnostiqueurs locaux tout en gardant les performances d'une structure centralisée. En effet, le procédé dispose de capteurs dit "globaux" qui doivent être dispatchés par projection pour avoir des observations locales sur les diagnostiqueurs locaux. Chaque diagnostiqueur local possède un sous-ensemble de l'ensemble des capteurs du procédé. En conséquence, un diagnostiqueur local observe uniquement une partie du procédé en utilisant une fonction de projection sur les événements qu'il observe.

Prenons l'exemple du modèle global du procédé de la Figure 2-16 [WAN 04]. Nous souhaitons réaliser une approche décentralisée où le modèle caractérise à la fois le comportement normal mais aussi le comportement anormal pour des défauts de type F1 et F2. A partir de ce modèle, il est possible d'établir deux diagnostiqueurs locaux qui vont dépendre

de leur observation locale. Le modèle G possède un ensemble d'observations $\Sigma_o = \{a_1, a_2, b_1, b_2, c_1, c_2\}$ et deux partitions de défaut $\Pi_{F1} = \{f_1\}$ et $\Pi_{F2} = \{f_2\}$. Chaque diagnostiqueur a sa propre observation : $\Sigma_{o1} = \{a_1, a_2, c_1, c_2\}$ pour G_{d1} et $\Sigma_{o2} = \{b_1, b_2, c_1, c_2\}$ pour le second diagnostiqueur local G_{d2}. Les diagnostiqueurs locaux obtenus sont représentés en Figure 2-17. G_{d1} est obtenu de la manière suivante :

- L'état initial du procédé G est inclus dans l'état initial du G_{d1} avec l'étiquette N, puisque l'état initial, 1, du G est supposé normal. Il en résulte 1N.
- On inclus tous les états du G atteignables par les événements non observables par G_{d1} dans son état initial avec leurs étiquettes correspondantes. Nous aurons 2F1, 3F1, 5F1, 6F2, 8N dus à l'occurrence de, respectivement, f_1, f_1, b_1, f_2, b_2.
- L'état initial du G_{d1} est représenté donc par {1N, 2F1, 3F1, 5F1, 6F2, 8N}.
- A partir de cet état du G_{d1}, on étudit tous les états atteignables par l'occurrence d'un événement observable par G_{d1}. Ainsi, avec l'occurrence de a_1, G_{d1} atteint le seul état 4 du G avec l'étiquette F1. C'est un état certain où G_{d1} peut décider avec certitude de l'occurrence de f_1.
- De la même manière, on construit le reste des états de G_{d1}.

La construction de G_{d2} s'effectue sur le même principe

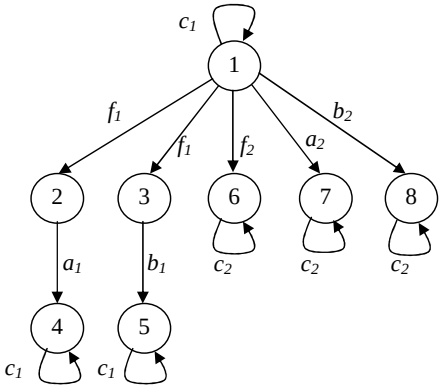


Figure 2-16 : Modèle du procédé G

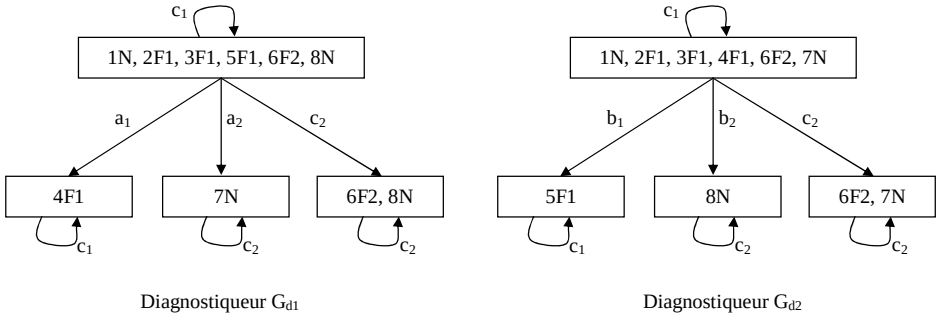


Figure 2-17 : Diagnostiqueurs locaux G_{d1} et G_{d2}

Chaque diagnostiqueur prend une décision locale. Cependant, les diagnostiqueurs locaux doivent fournir au final les mêmes performances qu'un diagnostiqueur global. L'observation de l'événement c_2 sur les deux diagnostiqueurs engendre une indécision sur le procédé. En

effet, après l'occurrence de c_2, G_{d1} retourne un état incertain {6F2, 8N} et G_{d2} un état {6F2, 7N} également incertain. Dès lors, il est impossible de prendre une décision sur le comportement du procédé. Si l'on regarde le diagnostiqueur global G_d (Figure 2-18) du procédé de la Figure 2-16, il en ressort une identification de tous les défauts et aucun état incertain. En effet, l'événement c_2 génère un défaut de type F2 sur l'état 6 du procédé. C'est pourquoi, un coordinateur de décisions locales doit être ajouté à la structure décentralisée afin de résoudre le problème d'indécision de l'occurrence du défaut de type F2.

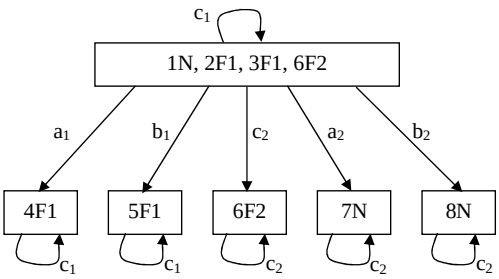


Figure 2-18 : Diagnostiqueur Global G_d

2.2.3.3. Structure décentralisée avec coordinateur

Afin de lever le problème d'indécision et permettre aux diagnostiqueurs locaux de diagnostiquer l'ensemble de défauts diagnosticables par un diagnostiqueur global, un coordinateur doit être utilisé. Il traite les différentes décisions locales, communiquées par les diagnostiqueurs locaux, afin de prendre une décision finale. Ce coordinateur peut être simple, à base de règles et sans mémoire [WAN 04] ou complexe, nécessitant l'estimation et la mémorisation de l'état du procédé [DEB 00].

L'obtention d'un coordinateur s'effectue par l'étude du modèle global du procédé par un expert. Il émet des priorités sur les décisions des diagnostiqueurs afin de lever les cas d'indécision. Le Tableau 2-4 représente un coordinateur simple pour le cas de deux diagnostiqueurs locaux [WAN 04]. Ce coordinateur est utilisé pour lever le problème d'indécision sur l'occurrence du défaut de type F2 de l'exemple de la Figure 2-17. Ce coordinateur prend la forme du Tableau 2-5.

Décision locale 1	Décision locale 2	Décision Globale
Défaut	Indécision	Défaut
Sans défaut	Indécision	Sans défaut
En défaut si aucun site "Sans défaut"	Indécision	Défaut
En défaut si aucun site "Sans défaut"	Défaut	Défaut
En défaut si aucun site "Sans défaut"	Sans défaut	Sans défaut
Sans défaut si aucun site en "Défaut"	Indécision	Sans défaut
Sans défaut si aucun site en "Défaut"	Défaut	Défaut
Sans défaut si aucun site en "Défaut"	Sans défaut	Sans défaut
Indécision	Indécision	Indécision
Défaut	Sans défaut	Conflit
En défaut si aucun site "Sans défaut"	Sans défaut si aucun site en "Défaut"	Conflit

Tableau 2-4 : Gestion des conflits décisionnels

En reprenant le cas d'indécision après l'événement c_2 pour la situation où G_{d1} atteint l'état {6F2, 8N} et G_{d2} l'état {6F2, 7N}, il est possible de prendre une décision finale à l'aide de la règle n°8 du coordinateur du Tableau 2-5. La décision finale est de considérer, dans ce cas, qu'un défaut de type F2 a eu lieu.

Règle :	Décision de G_{d1}	Décision de G_{d2}	Décision Globale
1	N, F1, F2	N, F1, F2	Indécision
2	F1	N, F1, F2	F1
3	N	N, F1, F2	N
4	N, F1, F2	F1	F1
5	N, F1, F2	N	N
6	F2, N	N	N
7	N	F2, N	N
8	F2, N	F2, N	F2

Tableau 2-5 : Table de décision du coordinateur de l'exemple de la Figure 2-17

Dans nos travaux, nous allons nous intéresser plus particulièrement à cette architecture décentralisée avec un coordinateur simple. Toutefois, les architectures décentralisées avec coordinateur complexe se retrouveront dans nos perspectives.

2.2.3.4. Structure distribuée

Le principe de la structure distribuée consiste à établir pour chaque composant du système son propre modèle associé à son diagnostiqueur local. A partir de ces différents sites, une interaction s'établit entre les modules communicants grâce à un protocole de communication [DAS 03a], [DAS 03b], [SU 04], [QIU 05]. Ce protocole définit la quantité et la destination de l'information à communiquer aux différents diagnostiqueurs locaux afin de lever tout problème d'indécision. Il n'est donc pas nécessaire de passer par un coordinateur de haut niveau. Les approches distribuées sont alors considérées comme un cas particulier des structures décentralisées où le cas d'indécision peut être levé grâce aux communications entre les diagnostiqueurs locaux.

La grande difficulté des approches distribuées est d'établir le protocole de communication entre les différents diagnostiqueurs. La complexité de ce protocole ne doit pas exploser avec l'ajout de nouveaux composants. De plus, le problème de délai de communication vient renforcer la difficulté de construction de se protocole [BOE 04], [DAS 03a], [DAS 03b]. C'est pourquoi, peu de méthodes sur le diagnostic distribué sont développées dans la littérature. Nous pouvons citer comme exemple de ces méthodes celles proposées en [HOL 94], [PAN 00], [FAB 02], [BEN 03], [SU 04], [BOU 04]. Nous ne développons pas cette structure dans ce mémoire de thèse. Elle fera l'aspect de perspectives de travaux par la suite.

2.2.3.5. Choix d'une structure

Le Tableau 2-6 rappelle les différentes caractéristiques que nous venons de citer pour chaque structure et fait ressortir, en grisé sur le tableau, tous les points qui nous intéressent.

	Centralisée	Décentralisée Sans Coordinateur	Décentralisée Avec Coordinateur	Distribuée
Modèle du procédé	Global	Global	Global	Local
Diagnostiqueur	Global	Local	Local	Local
Communication	Aucune	Aucune	Coordinateur	Entre modules
Décision	Globale	Locale	Locale	Locale
Indécision	Aucune	Risque important	Faible risque	Aucune
Complexité du Protocole	Aucune	Aucune	Faible	Importante

Tableau 2-6 : Etude comparative des différentes structures

Le choix d'une structure de décision pour le diagnostic s'effectue essentiellement selon le procédé et l'information disponible. Dans le cadre de notre étude, nous souhaitons nous focaliser sur les systèmes manufacturiers dont les éléments de PO fournissent une information décentralisée.

L'inconvénient des approches décentralisées avec coordinateur est le fait que le modèle du procédé, pour obtenir les diagnostiqueurs locaux, est global. Cela implique un risque d'explosion combinatoire dès la modélisation de la PO. Cependant, nous pensons que le coordinateur peut être enrichi afin d'exprimer uniquement les spécificités globales du procédé, non exprimées par les diagnostiqueurs locaux, sans pour autant décrire l'ensemble de son comportement.

L'approche décentralisée que nous allons proposer dans le chapitre 3 consiste à construire différents diagnostiqueurs locaux de manière totalement modulaire à partir de chaque élément du procédé et de prendre en compte les spécifications globales à travers le coordinateur. C'est dans cette optique que nous souhaitons développer notre approche afin d'adapter la structure décentralisée et la rendre moins sensible au problème d'explosion combinatoire au niveau du modèle du procédé.

2.2.4. Notion de diagnosticabilité

L'utilisation d'approches pour le diagnostic des défaillances est indispensable pour les systèmes plus ou moins complexes. Cependant, une question reste en suspend : le système est-il diagnosticable ? En effet, avant d'appliquer une méthode sur un système, il faut pouvoir vérifier si ce dernier dispose d'informations en quantité suffisante pour pouvoir effectuer le diagnostic. Par conséquent, la notion de « diagnosticabilité » va permettre de déterminer l'ensemble des pannes pouvant être diagnostiqué. Selon la structure (centralisée, décentralisée, distribuée) et l'information disponible, des extensions de la notion de diagnosticabilité ont été définies dans la littérature pour les SEDs.

Un SED est dit diagnosticable pour l'ensemble des partitions de défauts et pour un ensemble d'événements observables s'il est possible de détecter l'occurrence de n'importe quel défaut appartenant à une des partitions de défaut dans un délai fini [CON 04]. Cette

notion s'applique autant pour les méthodes à base d'états que celles à base d'événements. Dans ce paragraphe, nous présentons différentes notions associées à la modélisation à base d'événements et à base d'états, chacune dépendant de la structure utilisée pour le diagnostiqueur.

2.2.4.1. Formalisation de la notion de diagnosticabilité

2.2.4.1.1. Diagnosticabilité à base d'événements

La notion de diagnosticabilité à base d'événements peut être définie formellement de la manière suivante :

Définition 1 : Un langage L préfixe clos et vivant est dit diagnosticable par rapport à une fonction de projection P_L et un ensemble de partitions de défauts Σ_{Π} ssi [SAM 95] :

$$\forall f \in \Pi_{Fi},\ \forall i \in \{1, 2, \ldots, r\}, \exists\, n_i \in \aleph,\ \forall s \in \Psi(\Pi_{Fi}),\ \forall t \in L/s : |t| \geq n_i$$

$$\forall w \in P_L^{-1}(P_L(st)) \Rightarrow f \in w$$

où $L/s = \{t \in \Sigma^* \mid st \in L\}$ l'ensemble de toutes les séquences d'événements après s. $\Psi(\Pi_{Fi})$ est l'ensemble de toutes les séquences d'événements qui se termine par un événement de défaut appartenant à Π_{Fi}. $P_L^{-1}(P_L(st))$ correspond à l'ensemble de toutes les séquences d'événements qui ont une projection, une séquence observable d'événements, équivalente à celle de *st*.

Cette définition signifie qu'un langage L est diagnosticable si et seulement si, pour toute séquence de défaillance contenant un défaut appartenant à la partition Π_{Fi} des défauts de type Fi, le diagnostiqueur doit être capable d'isoler ce défaut après l'occurrence d'un nombre fini d'événement $n_i = |t|$ et que toute autre séquence *w* ayant un comportement observable, P(*w*), équivalent à celui de *st*, P(*st*), doit contenir un défaut appartenant à Π_{Fi}. Autrement dit, chaque défaut de l'ensemble des défauts Σ_f doit avoir une signature distincte et observable pour inférer l'occurrence de ce défaut et déterminer son type.

Après la construction du diagnostiqueur G_d et pour que le langage L soit diagnosticable, il suffit que ce diagnostiqueur satisfasse les deux conditions suivantes [SAM 95] :

1) Il existe au moins un état du diagnostiqueur pour lequel le diagnostiqueur décide avec certitude l'occurrence d'un défaut appartenant à une partition Π_{Fi}.
2) Il ne doit pas y avoir de cycles dits " indéterminés" pour lesquels le diagnostiqueur est incapable de décider avec certitude l'occurrence d'un défaut appartenant à une partition Π_{Fi}.

La Figure 2-19 présente un diagnostiqueur qui satisfait la condition 1 puisqu'il y a bien un état {6F1} pour lequel le diagnostiqueur décide avec certitude l'occurrence d'un défaut appartenant à Π_{F1}. Par contre, il ne vérifie pas la condition 2 puisqu'il y a un cycle *bcd* pour lequel le diagnostiqueur peut rester, jusqu'à l'infini, indécis de l'occurrence d'un défaut de type F1 ({3F1, 7N}, {4F1, 9F1, 11N}, {5F1, 10F1, 12N}). La présence de ce cycle indéterminé rend alors le système non diagnosticable [SAM 95]. De plus, si le modèle du procédé aboutit à un état bloquant où plus aucune évolution observable ne permet de détecter un défaut, alors le système est également non diagnosticable.

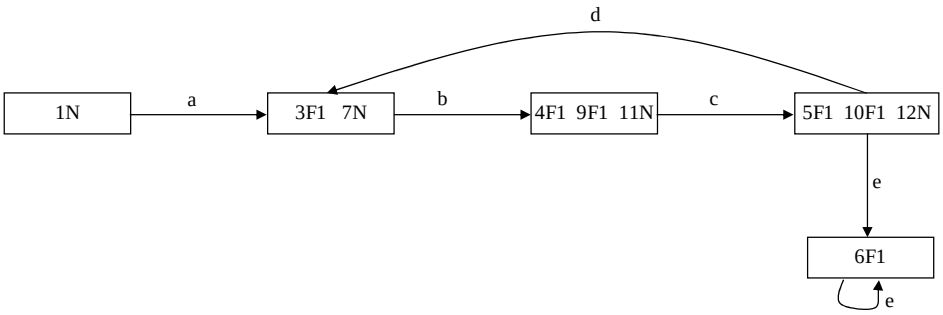


Figure 2-19 : Exemple d'un système avec un cycle F1-indéterminé dans son diagnostiqueur

La notion de diagnosticabilité a été également étendue en intégrant le temps de réaction du système [TRI 02]. Ainsi, il est possible de rendre compte du temps de réaction sur les actionneurs. Cette intégration du temps à la notion de diagnosticabilité permet notamment de distinguer les événements non observables des défauts, par une contrainte temporelle.

L'exemple de la Figure 2-20 montre un procédé permettant de distinguer un défaut *f* d'un événement non observable *u* par l'occurrence de l'observation *b* et par le temps de réponse de cet événement. Ainsi, à partir de l'état 2, l'événement *b* doit conduire dans l'état 4 ou 6 après un certains laps de temps. S'il survient après plus de 3 unités de temps, $t > 3$, il est alors possible de conclure que c'est le défaut *f* qui est survenu. Pour cet exemple, si l'information temporelle est absente, le procédé n'est pas diagnosticable.

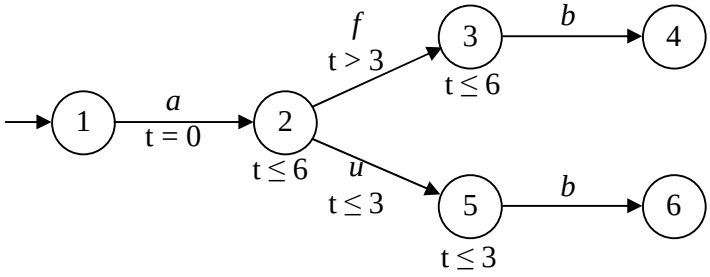


Figure 2-20 : Intégration du temps dans la notion de la diagnosticabilité

2.2.4.1.2. Diagnosticabilité à base d'états

La notion de diagnosticabilité pour les modèles à base d'état est définie dans [LIN 94] ou dans [MOU 05]. Cette notion peut être définie formellement de la manière suivante :

Définition 2 : Un système est diagnosticable pour un ensemble d'événements observables et un ensemble de partitions de défauts ssi :

$$\forall f \in \Pi_{Fi},\ \forall i \in \{1, 2, \ldots, r\}, \exists\, n_i \in \aleph,\ \forall s \in \Psi(\Pi_{Fi}),\ \forall t \in L/s : |t| \geq n_i$$

$$\forall x \in X,\ x' = \delta(st, x),\ x'' = \delta(w, x),\ \forall w \in P_L^{-1}(P_L(st))$$

$$h' = h(x'),\ h'' = h(x'') \Rightarrow h', h'' \in H_{Fi}$$

où H_{Fi} est l'ensemble des sorties de tous les états du diagnostiqueur atteints dus à l'occurrence d'un défaut de type Fi. Une sortie d'état d'un diagnostiqueur est caractérisée par les valeurs des capteurs de la PO et des ordres de la PC. Cette notion signifie que si l'état *x'* a été atteint

suite à l'occurrence d'une séquence d'événements *st* comprenant un défaut de type Fi et possédant une sortie $h' = h(x') \in H_{Fi}$, alors pour toute séquence d'événements *w* ayant un comportement observable équivalent à celui de st, $P_L(st) = P_L(w)$, doit faire évoluer le modèle vers un état *x''* qui a une sortie $h'' = h(x'')$ appartenant à H_{Fi}.

Prenons l'exemple du diagnostiqueur à base d'état de la Figure 2-14. Si on considère le seul défaut de type F indiquant que le radiateur est en panne, alors le modèle du procédé de la Figure 2-14a est diagnosticable. En effet, le diagnostiqueur peut atteindre l'état Bon, caractérisé par la sortie basse, B, du capteur de température et l'ordre *on* pour l'activation du radiateur, pour lequel il décide avec certitude l'occurrence du défaut de type F.

2.2.4.2. Co-diagnosticabilité

La notion de co-diagnosticabilité doit permettre d'assurer le fait que toute défaillance doit être diagnostiquée dans un délai borné par au moins un diagnostiqueur local en utilisant ses propres observations.

Définition 4 : Un langage L préfixe clos et vivant est dit F-co-diagnosticable par rapport aux fonctions de projection P_{Lj} ($j \in \{1, \ldots, m\}$) ssi [WAN 04] :

$$\forall f \in \Pi_{Fi},\ \forall i \in \{1, 2, \ldots, r\},\ \exists n_i \in \aleph,\ \forall s \in \Psi(\Pi_{Fi}),\ \forall st \in L,\ |t| \geq n_i$$

$$\exists j \in \{1, 2, \ldots, m\},\ \forall w \in P_{Lj}^{-1}(P_{Lj}(st)) \cap L \Rightarrow f \in w$$

où $P_{Lj}^{-1}(P_{Lj}(st)) \cap L$ est l'ensemble de toutes les séquences d'événements qui ont le même comportement observable par rapport au diagnostiqueur local G_{dj}. La F-co-diagnosticabilité signifie donc que n'importe quel défaut, appartenant à la partition Π_{Fi}, est diagnosticable par au moins un diagnostiqueur local G_{dj}.

Il est possible de définir la co-diagnosticabilité selon [QIU 04] qui considère que toute violation du langage de spécification K, représentant un cahier des charges, est due à un défaut :

Définition 5 : Un procédé composé de *m* diagnostiqueurs locaux avec un langage L préfixe clos et un langage de spécification K contenu dans L ($K \subseteq L$) est dit co-diagnosticable par rapport aux fonctions de projections P_{Lj} ($j \in \{1, \ldots, m\}$) ssi [QIU 04] :

$$(\exists n \in \aleph)\ (\forall s \in L\text{-}pr(K))\ (\forall st \in L\text{-}pr(K),\ |t| \geq n) \Rightarrow$$

$$\exists j \in \{1, 2, \ldots, m\},\ \forall w \in P_{Lj}^{-1}(P_{Lj}(st)) \cap L,\ w \in L\text{-}pr(K)$$

où *pr(K)* est le préfixe clos du langage *K* et représente le comportement désiré du procédé. Dès lors, pour toute séquence d'événements *s*, violant le cahier des charges et diagnosticable par un diagnostiqueur global après $n = |t|$ transitions de cette violation, il existe au moins un diagnostiqueur local, G_{dj}, capable de diagnostiquer cette violation en observant la séquence *st*. Il faut donc qu'après l'exécution d'au moins *n* transitions tous les défauts puissent être isolés par au moins un diagnostiqueur local pour que l'ensemble du procédé soit défini comme co-diagnosticable pour l'ensemble des partitions de défaut Σ_{Π}.

Reprenons l'exemple du procédé de la Figure 2-16 où les deux diagnostiqueurs, G_{d1} et G_{d2}, sont donnés en Figure 2-17. Dans le cas où les partitions de défauts sont $\Pi_{F1} = \{f_1\}$ et Π_{F2}

= $\{f_2\}$ alors le procédé est dit non co-diagnosticable. En effet, bien que les deux diagnostiqueurs permettent d'isoler les défauts de type F1, aucun ne permet d'isoler les défauts de type F2. Cependant, on peut dire que le système est F1-co-diagnosticable.

2.2.4.3. Co-diagnosticabilité forte

D'autres travaux ont développé la notion de co-diagnosticabilité pour garantir l'absence de défaut [WAN 04], [QIU 05]. On parle alors de co-diagnosticabilité forte (Strong-diagnosability). La NF-co-diagnosticabilité garantit alors l'absence d'un défaut par au moins un diagnostiqueur local.

Définition 6 : Un langage L vivant et préfixe clos est dit NF-co-diagnosticable par rapport aux fonctions de projections P_{Lj} ($j \in \{1, \ldots, m\}$) pour m sites locaux ssi [WAN 04], :

$$\exists n \in \aleph,\ \forall s \in L, \text{ où } s \text{ est une séquence sans défaut},\ \forall st \in L,\ |t| \geq n,$$

où st est une séquence sans défaut

$$\exists j \in \{1, 2, \ldots, m\},\ \forall w \in P_j^{-1}(P_j(st)) \cap L \Rightarrow w \text{ est une séquence sans défaut}$$

Cette définition permet de détecter l'absence de défaut et signifie que, pour une séquence d'événements s sans défaut et une séquence d'événements t suffisamment longue et sans défaut de continuité de s dans L, il existe au moins un diagnostiqueur local G_{dj} tel que toute séquence d'événements dans L non distinguable de st par le diagnostiqueur G_{dj} est aussi une séquence sans défaut.

Dès lors, il est possible de définir la co-diagnosticabilité forte selon [QIU 05] en intégrant le cahier des charges et en considérant que l'absence de défaut correspond à un comportement désiré défini par K.

Définition 7 : Un langage L préfixe clos et un langage de spécification K préfixe clos contenu dans L ($K \subseteq L$). Pour m sites locaux avec les fonctions de projections P_{Lj} ($j \in \{1, \ldots, m\}$), (L, K) est dit co-diagnosticable fort ssi [QIU 05] :

$$\exists n \in \aleph,\ \forall s \in pr(K),\ \forall st \in pr(K),\ |t| \geq n$$

$$\exists j \in \{1, 2, \ldots, m\},\ \forall w \in P_{Lj}^{-1}(P_{Lj}(st)) \cap L,\ w \in pr(K)$$

Il est possible de montrer qu'un procédé étant défini comme co-diagnosticable est alors également diagnosticable, la réciproque étant fausse [WAN 04].

2.2.4.4. Diagnosticabilité collaborative

La diagnosticabilité collaborative (Joint-Diagnosability) s'applique essentiellement aux structures distribuées [QIU 05]. C'est une extension de la co-diagnosticabilité puisqu'elle se base sur les informations locales de chaque diagnostiqueur mais également sur les informations des diagnostiqueurs voisins. En effet, la structure distribuée permet d'avoir une communication entre les différents diagnostiqueurs. Cette communication représente une information supplémentaire au niveau de chaque diagnostiqueur qui est nécessaire pour la prise de décision finale. Cette communication est définie par un protocole de communication [QIU 05].

Définition 8 : Soit un langage L préfixe clos et un langage de spécification K préfixe clos contenu dans L ($K \subseteq L$). Pour m sites locaux ($j \in \{1, \ldots, m\}$) communicant par un protocole *PR* à travers un canal FIFO (First In First Out) k-borné, (L, K) est dit diagnosticable collaboratif sous *PR* ssi [QIU 05] :

$$\exists n \in \aleph,\ \forall s \in L\text{-}K,\ \forall st \in L,\ |t| \geq n$$

$$\exists j \in \{1, 2, \ldots, m\},\ \forall w \in L,\ \Upsilon^{k}_{j}(st, w) = 1 \Rightarrow w \in L\text{-}K$$

où Υ^{k}_{j} est une fonction définie pour le protocole *PR* avec un retard de communication k-borné. Cette fonction détermine si les traces *st* et *w* peuvent être distinguées ou non par le diagnostiqueur local G_{dj}.

$$\Upsilon^{k}_{j}(st, w) = 1 \Rightarrow P_{Lj}(st) = P_{Lj}(w)$$

Dans cette notion de diagnosticabilité collaborative, *s* représente une séquence d'événements violant K, et *t* est une continuation bornée de cette séquence d'événements, $n = |t|$, nécessaire pour détecter cette violation par un diagnostiqueur global. Egalement, la séquence d'événements $w \in L$ est une séquence d'événements qui a un comportement observable, par le diagnostiqueur local G_{dj}, équivalent à celui de *st* après la communication des messages transmis par les autres diagnostiqueurs locaux sous le protocole *PR*. Cette notion signifie donc que *w* doit être également une séquence d'événements violant K.

2.3. Discussion sur les différents critères des méthodes de diagnostic des SED

Nous venons de voir les différents critères des méthodes de diagnostic issus de la littérature. Cette étude permet de justifier nos choix dans l'approche de diagnostic que nous allons développer.

Le Tableau 2-7 rappelle les différents outils de modélisation du comportement des procédés avec leurs avantages et inconvénients. Les automates à états nous semblent adaptés à la modélisation des systèmes manufacturiers pour le diagnostic des SED. En effet, les systèmes manufacturiers demandent une description précise de leur comportement d'un point de vue matériel (procédés à instance unique) et non pas d'un point de vue produit (procédés à instance multiple). De plus, l'approche de diagnostic, développée par la suite, demande une utilisation des outils de composition synchrone et de projection naturelle. Par ce fait, les automates à états permettent d'être manipulé grâce à la théorie des langages (voir Annexe 1). Les langages associés aux automates permettent également une définition formelle de la notion de co-diagnosticabilité.

Les automates à états seront utilisés comme base de modèle pour la constitution de nos diagnostiqueurs. Cependant, ceux-ci sont sujets au problème d'explosion combinatoire. Par conséquent, une modélisation modulaire sera réalisée. Une représentation des réactions des actionneurs en terme d'information temporelle nous semble également primordiale. Pour cela, nous utiliserons une modélisation des contraintes temporelles par des fonctions de prévision très proche des *templates*.

		Réf. :	Avantages	Inconvénients
Outils de modélisation	Automates à états	[SAM 95] [SU 04] …	Description précise, théorie des langages et outils de composition et de projection	Risque d'explosion
	RdPs	[GEN 03] …	Distribution des éléments	Risque d'explosion
	Expressions logiques	[WAN 00] …	Modèle abstrait	Interprétation parfois difficile
	Chroniques ou *templates*	[BOU 03a] [HOL 94] [PAN 00] …	Contraintes temporelles, procédé à instance unique et à instance multiple	Demande d'apprentissage sur les contraintes temporelles

Tableau 2-7 : Résumé des outils de modélisations

Les différents modèles de défauts, trouvés dans la littérature, sont basés uniquement sur les événements ou sur les états (Tableau 2-8). Afin de lever les inconvénients sur la détection des pannes intermittentes et des défauts à l'initialisation des procédés, nous souhaitons réaliser une modélisation des défauts mixte répondant à ces deux problèmes.

		Réf. :	Avantages	Inconvénients
Modèles de défauts	A base d'événements	[SAM 95] [SU 04]…	Pannes intermittentes	Défauts à l'initialisation
	A base d'états	[LIN 94] [ZAD 03] …	Défauts à l'initialisation	Pannes intermittentes
	Mixte	-	Détection des pannes intermittentes et des défauts à l'initialisation	

Tableau 2-8 : Résumé des modèles de défauts

En ce qui concerne la structure de la prise de décision, nous avons vu dans le Tableau 2-6 du paragraphe 2.2.3.5 que le choix d'une structure de décision pour le diagnostic s'effectue essentiellement selon le procédé et l'information disponible. Nous avons ciblé nos recherches sur une structure décentralisée avec coordinateur permettant de diminuer le risque d'explosion combinatoire récurrent aux structures centralisées. Les différents cas de conflits décisionnels seront gérer par un coordinateur à travers des règles et une table décisionnelle. Ainsi, il n'ay aura pas besoin d'un protocole de communication complexe entre les diagnostiqueurs locaux comme dans le cas des structures distribuées.

		Réf. :	Avantages	Inconvénients
Structure de la prise de décision	Centralisée	[SAM 95] …	Pas de communication	Explosion combinatoire
	Décentralisée sans coordinateur	[CHA 93] [WAN 04] …	Diminution de l'explosion	Risque de conflits décisionnels
	Décentralisée avec coordinateur	[DEB 00] [WAN 04] …	Diminution de l'explosion et gestion des conflits par le coordinateur	Nécessité d'un coordinateur
	Distribuée	[DAS 03a] [BOE 04] [SU 04] [QIU 05]...	Gestion des conflits directement	Protocole de communication

Tableau 2-9 : Résumé des structure de la prise de décision

Enfin, le Tableau 2-10 montre les différentes notions de diagnosticabilité. Ces notions vérifient la capacité d'un système à diagnostiquer une partition de défauts. Elles dépendent essentiellement de la structure de la prise de décision et de la modélisation des défauts. Par conséquent, comme nous avons opté pour une structure décentralisée avec une modélisation des défauts mixte, nous établirons par la suite une notion de co-diagnosticabilité mixte vérifiant les performances de nos diagnostiqueurs.

		Réf. :	Structure de la prise de décision
Capacité au diagnostic	Diagnosticabilité	[SAM 95] [TRI 02] [LIN 94] [MOU 05] …	Centralisée
	Co-diagnosticabilité	[QIU 04] [WAN 04] …	Décentralisée
	Co-diagnosticabilité forte	[QIU 05] [WAN 04] …	
	Diagnosticabilité collaborative	[QIU 05] …	Distribuée

Tableau 2-10 : Résumé sur les capacités au diagnostic

2.4. Positionnement de l'approche proposée

Nous avons développé une approche ayant pour objectif de diagnostiquer tout type de défauts pour les applications manufacturières. Basé sur une structure de décision décentralisée avec coordinateur (Tableau 2-6), cette démarche consiste à établir des diagnostiqueurs locaux à partir de modèles élémentaires de la PO et d'un modèle de spécifications en intégrant l'information temporelle des actionneurs.

L'approche permettant de construire, hors ligne, la structure décentralisée de diagnostic est

composée de six étapes et présentées succinctement ci-après. Une description détaillée de cette approche est fournie au chapitre 3.

1) La modélisation de chaque composant présent sur le système manufacturier en Elément de Partie Opérative EPO_i avec $i \in \{1, 2, \ldots, n\}$.
2) L'extraction du comportement désiré local de chaque EPO_i pour aboutir à des modèles d'EPO Commandés $EPOC_i$ avec $i \in \{1, 2, \ldots, n\}$.
3) L'intégration d'une information temporelle sur chaque $EPOC_i$ conduisant à des Eléments de Partie Opérative Commandés Temporels EPOCT.
4) L'ajout d'un état de défaut atteint suite à l'occurrence d'un défaut appartenant à la partition des défauts que l'on souhaite diagnostiquer. Il en résulte un $EPOCT_i$ étendu, avec $i \in \{1, 2, \ldots, n\}$, pour chaque élément i.
5) La constitution des diagnostiqueurs locaux D à partir des EPOCT étendu et d'un ensemble d'étiquettes indiquant la situation fonctionnelle de l'élément.
6) La réalisation du coordinateur C permettant la modélisation des contraintes de niveau global alors que les diagnostiqueurs n'expriment que des contraintes locales et la gestion des ambiguïtés et conflits décisionnels. Il permettra ainsi de prendre une décision finale DF lors de l'implantation en ligne.

Nous avons vu dans le chapitre 1 qu'il existait différentes approches d'implantation du diagnostic (par comparateur, par modèle de référence ou par filtre intercalé). Dans le cadre de nos travaux, nous avons souhaité réaliser la construction d'une approche de diagnostic mais également son implantation. L'implantation en ligne de l'approche de diagnostic proposée s'effectue par l'intégration des diagnostiqueurs et du coordinateur dans un filtre intercalé entre la PC implantée par l'utilisateur et la PO. Une description détaillée de cette approche est fournie dans le chapitre 3.

2.5. Conclusion

Ce chapitre a présenté un état de l'art des méthodes de diagnostic des défauts pour les Systèmes à Evénements Discrets. Ces méthodes ont été classifiées selon plusieurs critères : l'outils de modélisation (Automates, Réseaux de Petri, Expressions logiques), la modélisation des défauts (à base d'événements et à base d'états) et la structure de la prise de décision de diagnostic (centralisée, décentralisée sans coordinateur, décentralisée avec coordinateur et distribuée). Egalement, les différentes notions de diagnosticabilité, utilisées pour vérifier si chaque défaut appartenant à l'ensemble prédéfini des défauts est diagnosticable dans un temps fini, sont étudiées.

L'objectif de cet état de l'art est de justifier nos différents choix. Pour la réalisation de notre approche de diagnostic, nous avons besoin de manipuler nos différents modèles de la partie opérative et de la commande. Pour cela, nous avons choisis une modélisation par automates à états permettant l'utilisation de la théorie des langages.

L'information concernant le temps de réaction des actionneurs nous semble une information primordiale. Pour cela, l'utilisation de modèle temporel (à base de *templates* ou fonctions de prévision) nous paraît appropriée.

Dans ce chapitre, nous avons vu également différentes modélisations des défauts. Il en est ressorti de cette étude qu'une modélisation mixte (à base d'événements, à base d'états et à base d'informations temporelles) permet de répondre aux pannes intermittentes et aux défauts dus à l'initialisation des diagnostiqueurs.

Enfin, les différents travaux de la littérature sur la structure de la prise de décision ont permis de montrer que les structures centralisées étaient à éviter à cause de l'explosion du nombre d'états, que les structures distribuées permettaient de diminuer cette explosion mais imposaient des protocoles de communications parfois lourds et complexes pour la résolution des cas d'indécisions. Par la nature décentralisée de l'information présente sur les systèmes manufacturiers, nous avons opté pour une structure décentralisée permettant à la fois de diminuer le risque à l'explosion combinatoire et de régler les indécisions à l'aide d'un coordinateur simple d'agrégation des décisions.

L'approche de diagnostic des SED que nous proposons au chapitre 3 est donc une approche de diagnostic décentralisée avec coordinateur à base d'automates à états et de fonctions de prévision. Cette approche utilise une modélisation mixte des défauts qui nécessite donc une notion de co-diagnosticabilité mixte. Cette notion sera proposée et développée en chapitre 3. La Figure 2-21 représente les différents critères retenus pour notre approche de diagnostic.

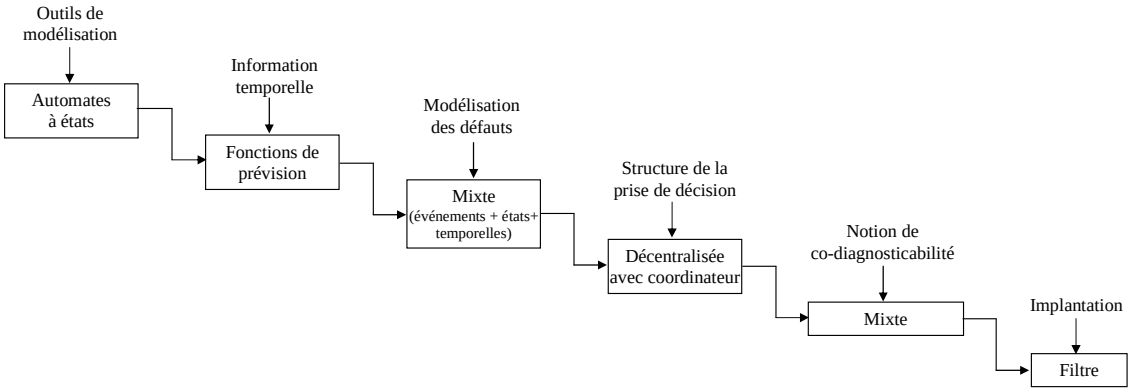


Figure 2-21 : Critères choisis de l'approche de diagnostic

Chapitre 3 : Diagnostic décentralisé des Systèmes à Evénements Discrets

3.1. Introduction

Pour répondre à la problématique présentée aux chapitres 1 et 2, nous avons développé une démarche de construction d'un diagnostic décentralisé pour les Systèmes à Evénements Discrets. Cette démarche globale s'effectue pour une part, hors ligne à travers la construction des diagnostiqueurs locaux et du coordinateur et, pour une autre part, en ligne pour son implantation. Nous n'avons pas souhaité restreindre notre étude à une partition de défauts particulière mais nous avons souhaité prendre en compte l'ensemble des possibilités de défaillances.

Avant de développer cette approche, il faut fixer au préalable quelques hypothèses de travail. En effet, nous souhaitons considérer le diagnostic des défaillances au niveau du matériel et non au niveau du produit. Par conséquent, les hypothèses suivantes sont posées :

- Seule la PO est représentée et non les produits (pièces, liquides, solides, …).
- L'inertie des actionneurs est considérée comme nulle au niveau de leur déplacement. De ce fait, si l'ordre d'arrêt est envoyé à un chariot, alors celui-ci s'arrête aussitôt et ne possède donc pas d'élan.
- On ne considère qu'un seul défaut possible sur chaque élément de PO. Par conséquent, si l'utilisateur observe deux défauts sur son système, alors ces deux défauts ne proviennent forcément pas du même élément de PO.

Ce chapitre présente la démarche de diagnostic décentralisé que nous avons développée [MOU 06], [PHI 05a], [PHI 05b], [PHI 05d], et dont les différentes étapes (Figure 3-1 et Figure 3-2) visent à prendre en compte (i) l'information de la PO, (ii) l'information de la PC et (iii) l'information temporelle liée aux temps de réaction des différents éléments de la PO. Dans un premier temps, nous présentons de manière conceptuelle chacune des six étapes. Ensuite, ces étapes sont détaillées sur un système de transfert de pièces.

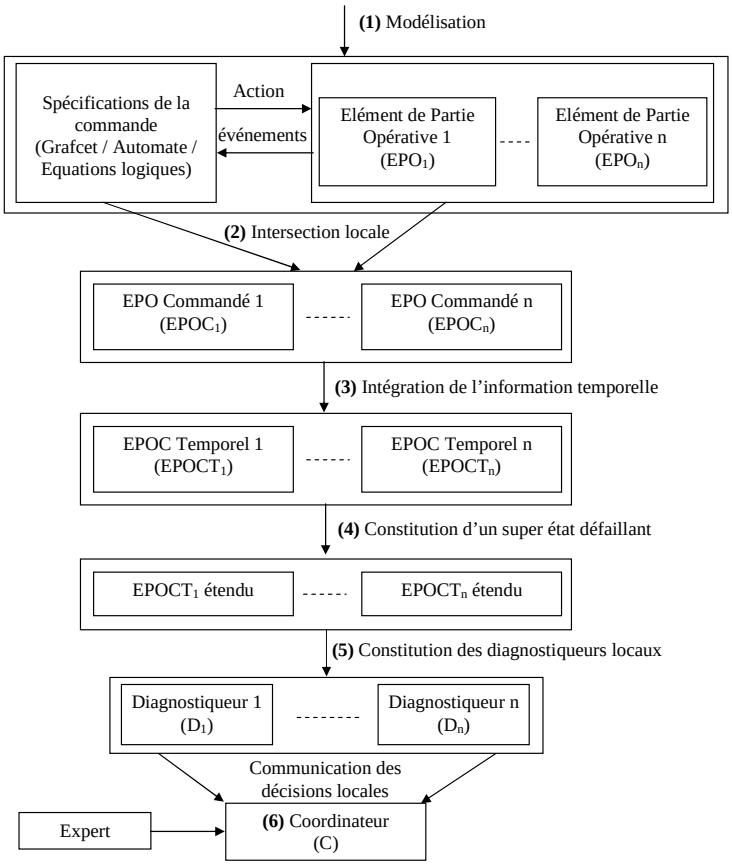


Figure 3-1 : Démarche globale de diagnostic décentralisé avec coordinateur

3.1.1. Présentation de la démarche hors ligne

1) Modélisation

L'étape de modélisation peut être composée en deux phases :

- L'élaboration des spécifications de la commande afin de représenter le comportement désiré du procédé. Ces spécifications sont représentées soit par une commande en GRAFCET considérée comme "optimale" en terme de sûreté de fonctionnement [NDJ 99], soit par des contraintes de vivacité et de sécurité exprimées sous forme d'automates ou d'expressions logiques.
- La modélisation de la partie opérative sous forme modulaire pour représenter le comportement logique de chaque élément par un modèle à base d'automate à états. Chacun de ces modèles est appelé un Elément de Partie Opérative : EPO_i avec $i \in \{1, 2, \ldots, n\}$ où n est le nombre d'éléments constituant la PO. Ces modèles (EPO) tiennent compte des spécifications technologiques de l'élément et de son fonctionnement élémentaire. Ce sont donc des modèles pratiques répondant à la réalité technologique des éléments dans les chaînes fonctionnelles des procédés manufacturiers. Ces modèles constituent une bibliothèque qui peut être utilisée pour modéliser le comportement de n'importe quelle PO de système manufacturier.

La modélisation de la PO s'effectue sous forme d'automates décrivant les évolutions physiquement possibles du procédé par des événements élémentaires. L'ensemble des

événements Σ peut être décomposé selon la notion d'observabilité. Ainsi, les événements liés aux capteurs et actionneurs d'un procédé pouvant être observés appartiendront à l'ensemble des événements observables $\Sigma_o \subseteq \Sigma$ alors que les événements de défauts ou n'appartenant pas à l'élément de PO seront considérés comme non observables $\Sigma_{uo} \subseteq \Sigma$. Cette notion d'observabilité est donc essentielle au diagnostic des défauts.

Une autre caractéristique importante pour un procédé est de pouvoir également distinguer les événements qui sont commandables de ceux qui ne le sont pas. En effet, les événements ne sont pas toujours générés de façon spontanée, mais réagissent à des commandes d'entrée, générant des réponses en sortie. Les interactions entre la PC spécifiée et la PO sont alors mises en évidence d'un point de vue commande. Par conséquent, nous adopterons par la suite que les ordres sont des événements envoyés par la PC et permettent de commander le procédé. Ils sont vu comme des sorties de la PC, ou comme les entrées de la PO, et sont des événements commandables $\Sigma_c \subseteq \Sigma_o$. A contrario, les événements provenant des capteurs répondent à une action en entrée de la PC, ou comme les sorties de la PO, et sont donc non commandables $\Sigma_{uc} \subseteq \Sigma_o$ [BAL 93]. Cette notion de commandabilité nous aidera notamment dans la construction des différents modèles de la PO constituant la base de notre approche de diagnostic.

Nous avons également retenu la correspondance suivante : un événement commandable correspond soit à l'activation, $\uparrow z$, soit à la désactivation, $\downarrow z$, d'un ordre z de la commande, tandis qu'un événement non commandable est associé soit au front montant, $\uparrow e$, soit au front descendant, $\downarrow e$, d'une variable d'entrée *e* de la PC. Un front montant correspond au passage de 0 à 1 d'un signal binaire, alors qu'un front descendant est considéré comme le passage de 1 à 0 de ce signal binaire. Les ensembles Σ_c et Σ_{uc} s'écrivent alors $\Sigma_c = \uparrow Z \cup \downarrow Z$ et $\Sigma_{uc} = \uparrow E \cup \downarrow E$, où $\uparrow Z$ et $\downarrow Z$ correspondent aux activations et désactivations de l'ensemble de tous les ordres de la PC et $\uparrow E$ et $\downarrow E$ reflètent, respectivement, les fronts montant et descendant de l'ensemble de tous les événements non commandables.

2) Intersection locale

Cette étape consiste à intégrer les informations du cahier des charges aux différents EPO. Elle se compose de deux étapes :

- La transformation de la spécification de la commande exprimée en GRAFCET en un Graphe Equivalent (*GE*) permettant d'avoir une sémantique identique à celle des modèles d'EPO.
- L'intersection locale entre le Graphe Equivalent et chacun des EPO afin d'obtenir le comportement désiré localement par les modèles d'EPO Commandés $EPOC_i$ ($i \in \{1, 2, \ldots, n\}$).

3) Intégration de l'information temporelle

L'étape d'intégration de l'information temporelle consiste à prendre en compte les contraintes temporelles qui s'exercent sur les EPOC. L'évaluation de ces contraintes est réalisée par des fonctions de prévision basées sur la logique floue. Cela signifie que chacune de ces fonctions peut prendre toutes les valeurs entre 0 et 1 et non pas simplement 1. Elle indique soit la violation d'une contrainte temporelle (valeur 1) soit le respect de la contrainte (valeur 0). L'intérêt de l'utilisation de la logique floue est de prévoir l'évolution vers un défaut par l'augmentation de la valeur fournie par ces fonctions de prévision de 0 vers 1. La

prise en compte de cette information temporelle aux différents EPOC conduit vers des modèles d'EPOC Temporels : $EPOCT_i$ ($i \in \{1, 2, \ldots, n\}$).

4) Constitution d'un super état défaillant

Cette étape consiste à créer un super état de défaut X_F aux $EPOCT_i$, $i \in \{1, 2, \ldots, n\}$. Ce super état de défaut est atteint soit par l'occurrence d'un événement non attendu, soit par la violation d'une fonction de prévision. Son rôle est de détecter les défauts susceptibles de survenir sans pour autant en diagnostiquer la cause. Il en résulte un modèle d'$EPOCT_i$ dit modèle EPOCT étendu, avec $i \in \{1, 2, \ldots, n\}$, pour chaque élément i.

5) Constitution des diagnostiqueurs locaux

Pour chaque $EPOCT_i$ étendu, un diagnostiqueur local D_i doit être créé. Ce diagnostiqueur doit permettre de localiser chaque défaut détecté et de l'identifier. L'ensemble des partitions de défauts est alors établi pour chaque diagnostiqueur local. A partir de chaque état de l'$EPOCT_i$ étendu, une fonction de décision comprenant une étiquette et indiquant la situation fonctionnelle du procédé est ajoutée.

6) Construction du coordinateur

Cette étape consiste à créer un coordinateur qui va gérer les cas d'indécision entre les diagnostiqueurs locaux, liés à l'observation partielle du procédé. Ce coordinateur est basé sur un ensemble de règles de fusion des décisions locales et un ensemble de règles permettant le diagnostic des défauts qui ne peuvent pas être diagnostiqués par au moins un des diagnostiqueurs locaux parce qu'ils nécessitent la prise en compte des interactions entre les diagnostiqueurs locaux. Ce coordinateur doit assurer un diagnostic équivalent à celui d'un diagnostiqueur global. Cela est vérifié par l'utilisation d'une notion de co-diagnosticabilité mixte, à base d'événements, à base d'états et à base d'informations temporelles, développée spécifiquement pour cette approche.

Les différentes opérations de compositions synchrones et de projections naturelles réalisées dans nos travaux, sont rappelées en annexe 1. Elles ont été effectuées à l'aide de l'outil Supremica, disponible librement pour l'éducation et la recherche sur le site http://www.supremica.org/.

3.1.2. Démarche en ligne

La construction des diagnostiqueurs locaux et du coordinateur représente un travail à réaliser en amont de manière hors ligne. L'aspect en ligne de l'approche proposée s'applique autour d'un filtre intercalé entre la PC et la PO très proche de celle proposée dans [LHO 91]. Ce filtre reçoit la décision finale provenant du coordinateur et des diagnostiqueurs locaux afin d'autoriser ou non soit l'envoi de l'ordre émis par la PC vers la PO soit des valeurs capteurs provenant de la PO à envoyer vers la PC (Figure 3-2). Le temps de réactivité entre les ordres envoyés par la PC à la PO et les valeurs des capteurs fournies par la PO à la PC se retrouve alors augmenté par le filtre. Pour notre étude, ce retard de communication sera considéré comme négligeable.

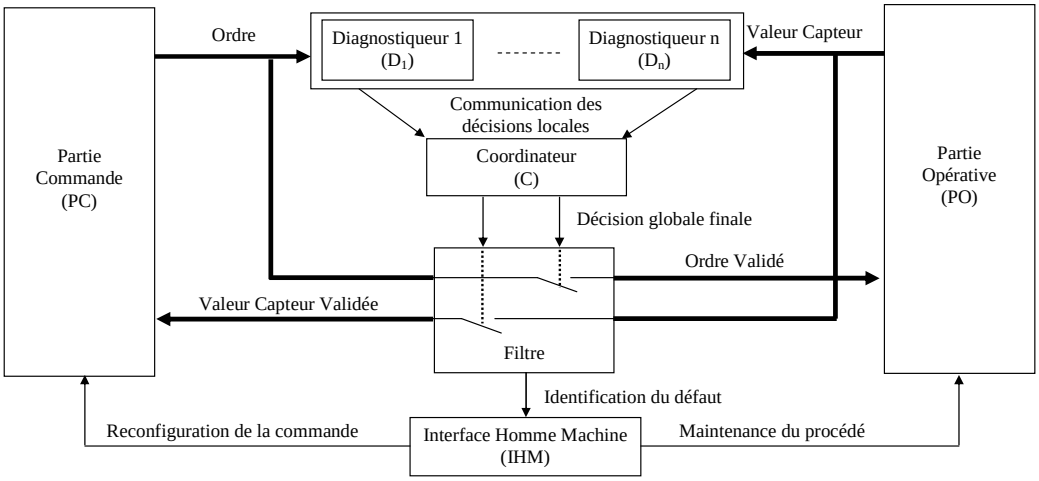


Figure 3-2 : Implantation en ligne de l'approche de diagnostic proposée dans la thèse

Chaque ordre provenant de la PC est envoyé dans le filtre à l'ensemble des diagnostiqueurs locaux et au coordinateur avant d'être envoyé vers la PO. Le coordinateur récupère et fusionne ensuite l'ensemble des décisions locales afin de rendre une décision globale finale autorisant ou non l'envoi de l'ordre validé. Sur le même principe, chaque valeur de capteur provenant de la PO va passer dans les diagnostiqueurs locaux et le coordinateur. La décision finale, sortant du coordinateur, valide la valeur du capteur que va recevoir la PC. En cas de défaillance, le module de décision renvoie l'étiquette du défaut à l'opérateur humain à travers une Interface Homme Machine (IHM) afin qu'il puisse identifier les défauts et répondre à celui-ci par une reconfiguration de la commande ou une intervention de maintenance sur le procédé.

3.2. Modélisation des éléments de la partie opérative (EPO)

La modélisation d'un système manufacturier est très souvent utilisée pour faire ressortir les caractéristiques de ce dernier. Selon l'objectif de l'étude, ce modèle sera plus ou moins détaillé et enrichi en fonction de l'outil qu'il lui est associé [CAR 03], [GOU 03], [ROU 05], [KLE 05]. En effet, si l'intérêt ne porte que sur une évolution globale du procédé, le modèle n'a pas forcément besoin d'une description précise et détaillée mais simplement d'une représentation de ces situations. Ainsi, nous souhaitons modéliser chaque événement élémentaire associé à un composant et pas seulement son état de fonctionnement afin d'aider au diagnostic de tous les défauts possibles. Par exemple, un vérin ne sera pas modélisé par un état de sortie et un état de rentrée, mais bien par tous les événements exprimant son déplacement comme le fait de quitter la position de rentrée pour se retrouver dans un état intermédiaire avant d'être en position de sortie complète.

La description précise du comportement de la PO est une opération complexe car les évolutions d'un système physique sont de nature asynchrone et non déterministe. Asynchrone dans le sens où chaque élément de la PO peut évoluer librement et où la probabilité que deux événements surviennent au même instant est quasiment nulle. Non déterministe dans le sens où pour une même séquence d'événements envoyé par la PC, la PO peut évoluer de manières différentes et ne pas atteindre la même situation, et ceci notamment dans les cas d'usure et de vieillissement du matériel.

Les modèles de PO doivent tenir compte à la fois de la technologie du matériel utilisé, et de l'environnement du procédé. Pour contourner les difficultés d'une modélisation globale, nous avons choisi une démarche modulaire permettant d'exprimer des causalités simples entre les éléments de la PO. Par conséquent, la modélisation s'effectue à l'aide d'un automate réceptif aux ordres de la partie commande et qui agit sur l'évolution des valeurs logiques de ses entrées de manière à exprimer les réactions de la PO. Comme pour tout modèle, les entrées/sorties du modèle du procédé à commander sont fixées par l'utilisation finale de ces modèles. Les entrées du procédé à commander sont les sorties du contrôleur industriel tandis que les sorties du procédé à commander sont les entrées du contrôleur industriel.

Le comportement de chaque modèle de PO peut être décrit par un automate à états dont l'élaboration correspond à un réel travail d'expertise nécessitant :

- une connaissance détaillée de la technologie mise en œuvre. Quels sont les moteurs, les vérins et les distributeurs mis en place sur le système ?
- la maîtrise de l'outil de modélisation employé,
- une connaissance de l'utilisation finale du modèle. Par exemple, il faut savoir comment réagit un vérin double effet associé à un distributeur 5/2 bistable par rapport à un distributeur 5/3 Centre Ouvert.

L'objectif de ce paragraphe est d'utiliser l'information présente sur la PO comme information de base pour le diagnostic des défaillances. Pour cela, nous développons trois méthodes d'obtention de modèles de bas niveaux de la PO (intuitive, théorique et pratique) afin de décrire le comportement du procédé dans ces moindres détails. Nous souhaitons montrer ici l'influence de la construction sur la richesse des modèles de la PO. Dans un premier temps, nous verrons que la plupart des modèles de la PO sont obtenus par construction intuitive à partir des connaissances du procédé mais surtout grâce à l'expérience acquise. Cependant, l'expert peut se tromper dans son modèle. Nous présentons ensuite une construction théorique structurée du modèle de la PO à travers des règles d'évolution afin d'aider l'expert dans sa modélisation. Néanmoins, ces modèles répondent à des règles logiques, donc théoriques, qui ne correspondent pas toujours à la réalité. Il apparaît alors un écart de comportement entre le modèle théorique et le modèle pratique. Une troisième construction du modèle de PO est par conséquent introduite à travers une construction pratique des modèles de PO afin d'établir au final une bibliothèque d'éléments de la PO qui sera validée par un expert.

Nous insistons sur le fait qu'un modèle de PO doit représenter l'ensemble des situations possibles et ne pas prendre en compte la moindre contrainte provenant le la PC. Par conséquent, les différents modèles que nous allons présenter vont présenter certaines situations aberrantes mais réalisables. Par exemple, l'envoi de deux ordres opposables semble illogique d'un point de vue sécurité, mais rien n'empêche l'utilisateur d'effectuer cette erreur. Nous avons pris le parti de modéliser la PO de façon totalement indépendante de la commande. Par conséquent, aucune contrainte de sécurité ou de vivacité ne doit être introduite dans nos modèles. L'intégration de ces contraintes fera l'objet d'une discussion dans le paragraphe 3.3.3.

Afin de pouvoir comparer chaque modélisation (intuitive, théorique et pratique), nous allons étudier la construction d'un modèle de Vérin Double Effet (VDE) piloté par un distributeur pneumatique 5/2 bistable muni de 2 détecteurs de fin course *a* et *b* (Figure 3-3). Le vérin est commandé à la fois en sortie par l'ordre *SO* et en rentrée par l'ordre *RE*. La

technologie du distributeur associée au vérin engendre quelques spécificités qui devront être représentées :

- Une activation puis désactivation d'un même ordre ne change pas la position des chambres du distributeur mais implique une réaction du vérin. Ainsi, une activation de l'ordre *SO* déplace le tiroir du distributeur vers la droite. Si maintenant on désactive ce même ordre alors le tiroir du distributeur ne bouge pas et entraîne quand même une sortie du vérin.
- Une priorité sur le premier ordre envoyé existe en cas de conflit de commande. Cela signifie que si l'ordre de sortie *SO* est envoyé avant l'ordre de rentrée *RE*, alors le vérin effectuera une sortie de sa tige.

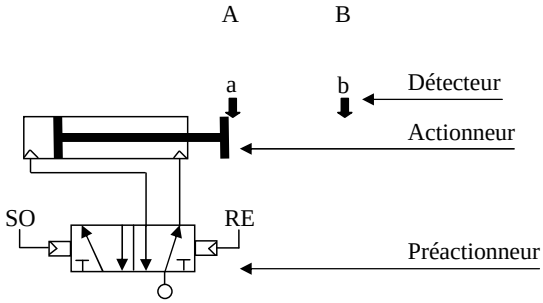


Figure 3-3 : Vérin double effet piloté par un distributeur 5/2 bistable

D'autres spécificités technologiques, par exemple avec une bobine du distributeur avec un différentiel permettant de donner toujours la priorité à un ordre, existent. Cependant, nous ne les verrons pas au cours de cette étude. Les modèles des éléments de partie opératives les plus utilisés dans les systèmes manufacturiers sont détaillés dans l'annexe 2.

3.2.1. Modélisation intuitive

La modélisation intuitive consiste à construire le modèle à partir des connaissances de l'expert sur le matériel. Il n'existe donc pas de méthode mais simplement une logique de construction à appliquer en partant d'une situation bien connue et en essayant d'imager toutes les évolutions possibles.

Prenons l'exemple de la Figure 3-4 décrivant le comportement du VDE de la Figure 3-3. En position initiale (état 0), le vérin peut se déplacer soit à droite (état 1), soit à gauche (état 5). L'activation de l'ordre *SO* lance le déplacement du préhenseur vers la droite, lui faisant quitter la position A par l'événement $\downarrow a$ (état 2). La continuation du mouvement lui fait atteindre le détecteur *b* (état 3). La désactivation de l'ordre *SO* lui permet de revenir à l'état initial. A partir de l'état 1, si l'ordre *SO* est désactivé, le vérin retourne dans son état initial. Dans l'état 2, la désactivation de *SO* ne permet plus un retour en situation initiale car la technologie utilisée force le préhenseur à aller vers la droite (état 4 puis état 0). Si après la désactivation de *SO* (état 4), l'ordre de rentrée *RE* est envoyé (état 6), alors le vérin retourne en position A par l'activation du détecteur *a* (état 7). La désactivation de l'ordre *RE* permet un retour en état initial. Le comportement du vérin pour un mouvement vers la gauche à partir de la situation initiale est similaire à celui de droite.

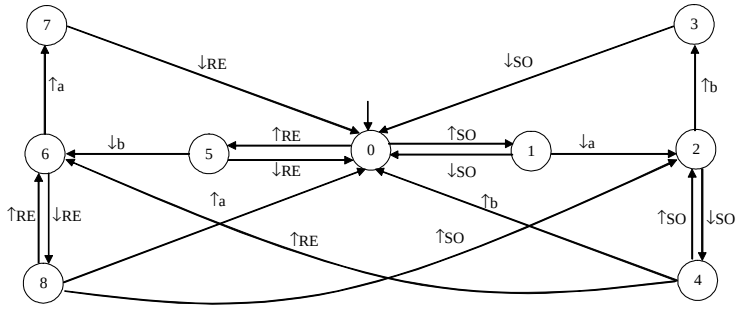


Figure 3-4 : Modèle du vérin

Ce modèle, construit intuitivement, semble décrire les différentes séquences de fonctionnement du vérin. Cependant, en regardant de plus près, il est possible de s'apercevoir que ce modèle ne décrit pas le vérin dans la totalité de ses situations. Plusieurs incohérences existent comme le fait que pour un même état 0, le vérin est soit en position de sortie soit en position de rentrée. De même, la séquence permettant la sortie du vérin est réalisable plusieurs fois, ce qui n'est pas envisageable réellement. De plus, certaines contraintes décrivant la commande sont déjà prises en compte alors qu'elles n'ont pas lieu d'être. Dès lors, la modélisation intuitive représente un travail complexe et une source d'erreurs très souvent inévitables pour l'expert.

Les modèles d'éléments de PO sont souvent très complexes et conduisent à se poser plusieurs questions : Comment être sûr d'avoir bien décrit la PO du système sans y avoir intégré des contraintes ou fait de redondance de situation ? Comment vérifier que rien n'a été oublié en termes d'états et de transitions ? Cette modélisation fait appel non seulement à une bonne connaissance du procédé, mais également à l'expérience et aux compétences en terme de modélisation du concepteur.

Lors de l'élaboration des modèles du procédé et de ses spécifications, le concepteur a tendance à faire l'amalgame entre le modèle de la PC, le modèle de la PO et le modèle des contraintes de sécurité et vivacité [PHI 03]. Par exemple, au niveau de la commande, le concepteur cherche implicitement à intégrer une image de la PO afin d'exprimer la causalité entre l'envoi des ordres et les réactions conséquentes de la PO. Ceci suppose cependant, que toutes les réactions du procédé et leur ordonnancement, sont connus de manière précise pour chaque action et chaque situation de la commande. Cette hypothèse est irréaliste lorsque l'on sait que la complexité et la nature parallèle des systèmes de commande et des procédés à commander entraînent des évolutions parallèles et complexes entre l'envoi des ordres et les réactions conséquentes du procédé. De même au niveau de la PO, ce sont les contraintes de sécurité que le concepteur tente d'intégrer implicitement afin d'exprimer au plus tôt, des restrictions sur le comportement du procédé.

Dans l'exemple de la Figure 3-3, le modèle du vérin intègre la contrainte *« Ne pas sortir et rentrer en même temps »*. La contrainte est simple à exprimer car il s'agit d'un modèle élémentaire de PO mais ceci risque d'être beaucoup plus complexe lorsqu'il s'agit de modèles où la granularité du comportement modélisé est différente. D'autre part, quand des éléments du procédé doivent être ajoutés, retirés ou configurés différemment, le concepteur doit revoir l'ensemble des modèles et ses interactions. Il est donc préférable pour la PO de s'appuyer sur des modèles explicites et indépendants de la commande. Néanmoins, la modélisation explicite de la PO est une tâche complexe qui se heurte non seulement à des problèmes méthodologiques de structuration et de composition de ses éléments mais aussi au choix de la

granularité du comportement modélisé. Par conséquent, nous proposons par la suite une méthode structurée à base de règles, en s'appuyant sur des travaux de Chandra [CHA 01], pour aider le concepteur dans sa tâche de conception des modèles de PO.

3.2.2. Modélisation théorique structurée

Pour structurer la modélisation de la PO, nous avons retenu une modélisation à base de règles définissant les interactions entre les événements commandables et les événements non commandables ; et des relations précisant les liens entre les événements non commandables. Il s'agit de règles dites d'occurrence et dans le second cas, de relations de précédence [PHI 04a], [PHI 04b]. Ces règles définies par l'utilisateur sont ensuite traduites en automates compatibles avec les principes énoncés précédemment tout en conservant la sémantique et l'interprétation de Balemi [BAL 93].

a) Principe

Avant de déterminer les règles d'occurrence des événements commandables, il faut commencer par fixer le contexte c'est à dire les conditions initiales du système. Il s'agit ensuite de recenser tous les événements liés à l'élément de PO que l'on cherche à modéliser puis de définir sous forme de règles, l'influence de l'activation et de la désactivation des événements commandables sur les événements non commandables. Par conséquent, chaque règle va s'exprimer selon le principe simple d'une « cause/conséquence », la cause étant liée à l'événement commandable et la conséquence portant sur l'événement non commandable. Pour chaque règle d'occurrence ayant la même cause, il faut ensuite établir sous forme de relations de précédence, la chronologie liant les événements non commandables conséquents.

L'obtention de ces informations s'effectue de la manière suivante :

- Définition des conditions initiales sur la position des détecteurs et sur les ordres envoyés par la PC. Ce sont donc les valeurs initiales des événements commandables $\sigma \in \Sigma_c$ et non commandables $\alpha \in \Sigma_{uc}$ qui doivent être définis dans cette étape.
- Interaction entre les sorties et les entrées du système afin de déterminer des règles d'occurrence de type cause/conséquence. Ainsi, s'il existe un événement non commandable $\alpha \in \Sigma_{uc}$ qui est conséquence d'un événement commandable $\sigma \in \Sigma_c$ alors une règle d'occurrence est définie par :

$$\sigma \rightarrow \alpha \quad avec \begin{cases} \sigma = \uparrow z \; ou \downarrow z \\ \alpha = \uparrow e \; ou \downarrow e \end{cases}$$

"$\rightarrow$" signifiant qu'après l'événement σ, l'événement α "finira par avoir lieu".

- Etablissement de la chronologie des événements non commandables pour les règles ayant la même cause par des relations de précédence. Cette séquence d'événements non commandables fait intervenir l'opérateur humain car il n'y a aucune logique permettant de valider ces relations. Les différentes relations de précédence sont établies par :

$$\forall \sigma \in \sum_c \; et \; \forall \alpha_1, \alpha_2 \in \sum_u \; avec \; \sigma \rightarrow \alpha_1 \; et \; \sigma \rightarrow \alpha_2$$
$$alors \; \alpha_1 \Rightarrow \alpha_2 \; (ou \; \alpha_2 \Rightarrow \alpha_1)$$

"⇒" signifiant que l'événement α_1 "précède" l'événement α_2.

A partir des règles d'occurrence et des relations de précédence, un modèle automate équivalent est obtenu par construction automatique selon les quatre étapes suivantes :

1) Construction de l'automate G_c à 2^n états (n étant le nombre de sorties de la commande) décrivant toutes les évolutions possibles des événements commandables à partir des conditions initiales données.

2) Constitution de l'automate d'occurrence G_{oc} à partir de l'automate G_c obtenu précédemment. Cet automate est obtenu par ajout sur chaque état des transitions rebouclantes correspondant aux événements non commandables résultant des règles d'occurrence. Dès lors, un état peut autoriser à la fois l'activation et la désactivation d'une entrée.

3) Construction de l'automate dit de précédence G_{pre} à partir des relations de précédence et de la situation initiale des événements non commandables.

4) Constitution de l'automate final $G = (X, \Sigma, \delta, x_0, X_m)$ par composition synchrone de l'automate issu des règles d'occurrence et de l'automate de précédence $G=G_{oc}||G_{pre}$.

b) Application

Dans le cadre de l'exemple du vérin de la Figure 3-3, il existe deux ordres *SO* et *RE*, et deux détecteurs *a* et *b*. En situation initiale, le vérin se trouve en position A et aucun ordre n'est envoyé à la PO. Afin de définir les règles d'occurrence, il faut s'intéresser aux conséquences des ordres envoyés. L'activation de l'ordre *SO* permet au vérin de se déplacer vers la droite et de quitter le détecteur *a*. De même, l'activation de *RE* fait revenir le vérin de la position B jusqu'à la position A par l'activation du détecteur *a*. Suite à l'activation de *SO*, la chronologie des événements non commandables est la suivante : désactivation de *a* puis activation de *b*. Pour la relation de précédence liée à l'activation de l'ordre *RE*, la chronologie se présente ainsi : désactivation de *b* avant activation de *a*. La désactivation de l'ordre *SO* engendre également le déplacement du vérin comme le précise la spécificité 1. Dès lors, les règles d'occurrence sur cet ordre aboutissent à la relation de précédence définissant la désactivation de *a* avant activation de *b*. Les conditions initiales et les différentes relations sont résumées sur le Tableau 3-1.

Conditions Initiales	Règles d'occurrence	Relations de précédence
SO = 0 *RE* = 0 *b* = 0 *a* = 1	$\uparrow SO \rightarrow \downarrow a$ $\uparrow SO \rightarrow \uparrow b$	$\downarrow a \Rightarrow \uparrow b$
	$\uparrow RE \rightarrow \downarrow b$ $\uparrow RE \rightarrow \uparrow a$	$\downarrow b \Rightarrow \uparrow a$
	$\downarrow SO \rightarrow \downarrow a$ $\downarrow SO \rightarrow \uparrow b$	$\downarrow a \Rightarrow \uparrow b$
	$\downarrow RE \rightarrow \downarrow b$ $\downarrow RE \rightarrow \uparrow a$	$\downarrow b \Rightarrow \uparrow a$

Tableau 3-1 : Interactions et relations entre événements

Pour l'exemple du VDE de la Figure 3-3, l'étape 1 de cette méthode consiste à établir pour le vérin un automate G_c à 4 états avec des évolutions entre les états liées à l'activation ou la désactivation de l'ordre *SO* et de l'ordre *RE*. Aucun ordre n'étant activé dans les conditions initiales, l'état 0 de la Figure 3-5a permet soit l'activation de l'ordre *SO* (état 1), soit

l'activation de l'ordre *RE* (état 2). A partir de l'état 1, la désactivation de *SO* est alors possible ce qui permet de retourner à l'état initial. A partir de l'état 1, l'activation de l'ordre *RE* est également possible et permet d'atteindre l'état 3. De même, à partir de l'état 2, il est possible de retourner à l'état 0 par désactivation de *RE* ou de partir vers l'état 3 par activation de *SO*.

A partir de cet automate, l'étape 2 consiste à ajouter des boucles sur les états prenant en compte les événements non commandables définis dans les règles d'occurrence. L'automate G_c est complété par les événements non commandables *a* et *b* en respectant les règles d'occurrence définies par l'utilisateur afin d'obtenir l'automate de règles d'occurrence G_{oc} (Figure 3-5b). Suite à l'activation de l'ordre *SO* (états 1 et 3), il est possible de désactiver le détecteur *a* et d'activer *b*. L'activation de l'ordre *RE* entraîne quant à lui des boucles dans les états 2 et 3 avec l'activation de *a* et la désactivation de *b* comme événements non commandables.

L'étape 3 conduit à établir avec les deux relations de précédence liant *a* et *b*, l'automate de précédence G_{pre} illustré Figure 3-5c. A l'initialisation, le vérin est en position A (état 0), l'activation de *b* (état 2) ne peut se faire qu'après une désactivation du détecteur *a* en état 1. De même, pour retourner en état 0, la désactivation de *b* (état 1) précède l'activation de *a*. Les ordres de la commande pouvant être envoyés à n'importe quel instant, cet automate autorise alors tous les événements commandables Σ_c du système à travers des transitions rebouclantes sur chaque état. Cette autorisation de l'ensemble des événements commandables est indispensable pour l'obtention du modèle de PO final.

Pour obtenir l'automate final *G* du vérin (Figure 3-5d), il suffit alors d'effectuer une composition synchrone entre l'automate d'occurrence G_{oc} issu de l'étape 2 et l'automate de précédence G_{pre} issu de l'étape 3. Le vérin est alors décrit de façon complète indépendamment de la commande à travers un automate à 12 états et 35 transitions. La comparaison avec le modèle intuitif montre clairement qu'aucune contrainte n'a été prise en compte dans le modèle de la Figure 3-5d, car par exemple, pour l'état 3, les deux ordres *SO* et *RE* sont envoyés en même temps à la PO.

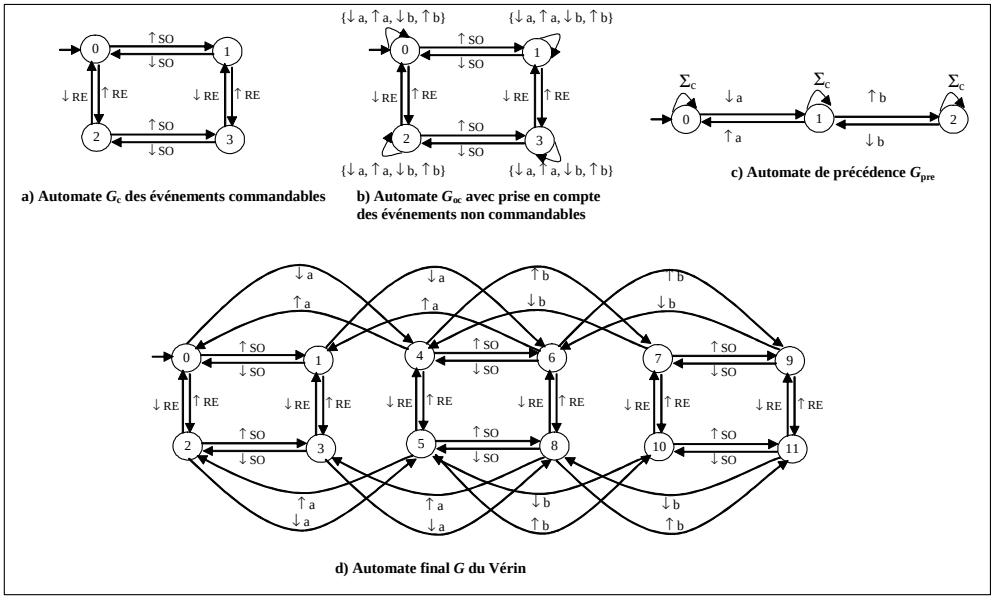


Figure 3-5 : Construction structurée du modèle théorique du vérin

3.2.3. Modélisation pratique

Nous venons de voir une modélisation théorique et structurée d'un EPO. Cette modélisation est structurée dans le sens où la construction des modèles se fait de manière logique, appliquée et non plus uniquement intuitive. Cependant, le fait de construire ce modèle automatiquement implique des réactions logiques d'un point de vue théorique qui ne sont plus valables dans la réalité. En effet, chaque EPO dispose de spécifications technologiques qui vont différencier le modèle théorique du modèle pratique. La modélisation pratique constitue alors un compromis idéal entre la modélisation intuitive et théorique. Elle fait appel à la fois à une structure classique et à la connaissance technologique d'un expert.

La modélisation pratique d'un EPO se base sur les spécifications technologiques du composant et de son fonctionnement élémentaire. Pour connaître ces caractéristiques, il faut pouvoir étudier la chaîne fonctionnelle d'un procédé. La chaîne fonctionnelle d'un procédé (Figure 3-6) permet d'envoyer des ordres à la PO à partir de la PC par une chaîne d'actions, et de recevoir une information sur la réaction de la PO par une chaîne d'acquisition. La chaîne d'actions est composée de préactionneurs permettant de gérer l'énergie provenant de la PC. Cette énergie est ensuite envoyée aux actionneurs afin de la convertir en action sur l'effecteur. La chaîne d'acquisition permet à travers des détecteurs, voire des transmetteurs, d'acquérir et de transmettre l'information de l'effecteur.

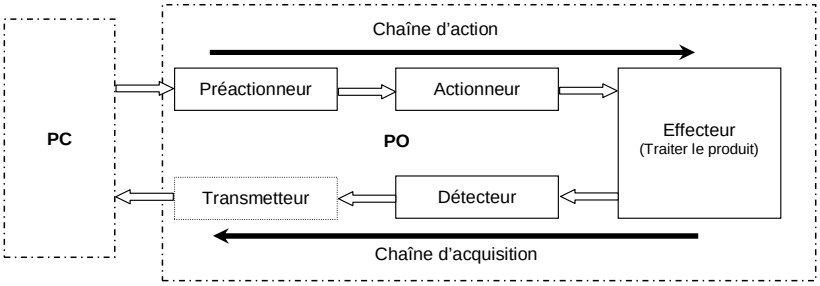


Figure 3-6 : Chaîne fonctionnelle d'un procédé

Les instructions provenant de la commande sont, par conséquent, envoyées aux préactionneurs qui vont permettre l'alimentation en énergie des actionneurs. Les détecteurs réagissent alors aux différentes actions reçues par les actionneurs. La modélisation d'un Elément de Partie Opérative (EPO) consiste donc à représenter cette chaîne fonctionnelle à travers ces éléments que sont : les préactionneurs, les actionneurs et les détecteurs.

Les différents composants élémentaires d'une PO sont obtenus en fonction de leur technologie. Chaque modèle de chaque composant est défini en deux parties :

- Le modèle des détecteurs lié aux entrées de la PC,
- Les modèles préactionneurs et actionneurs liés aux sorties de la PC.

Le modèle des détecteurs est défini par un automate G_{det} et dépend essentiellement du nombre de détecteurs présents. Le préactionneur est modélisé par un automate G_{preac}. Il est défini en fonction de la technologie : Distributeur pneumatique 5/2 monostable, 5/2 bistable, 5/3 bistable centre ouvert, 5/3 bistable centre fermé, contacteur moteur électrique à 2 sens de rotation .... L'automate de l'actionneur G_{act} s'obtient en combinant l'automate du préactionneur avec les événements du détecteur autorisés par le procédé. Au final, le modèle

pratique de la PO_i est un automate G_i obtenu par composition synchrone entre le modèle des détecteurs et le modèle actionneur (Figure 3-7).

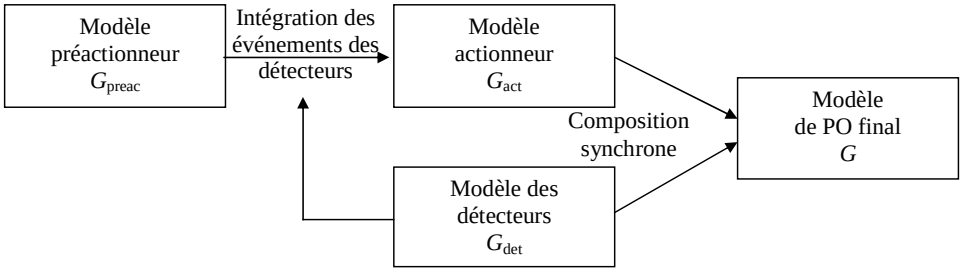


Figure 3-7 : Obtention du modèle pratique de PO

3.2.3.1. Modèle des détecteurs

Le modèle des détecteurs représente les situations du système en fonction de ses détecteurs. Il est donc lié aux entrées de la PC. Si l'on considère un système à deux détecteurs, le modèle est représenté alors par un automate à 3 états $G_{det} = (X_{det}, \Sigma_{uo}, \delta)$ (Figure 3-8). Celui-ci représente alors les situations où le système se trouve en position A (état 0), en position B (état 2) et entre les deux détecteurs (état 1).

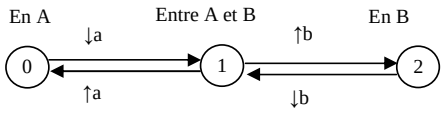


Figure 3-8 : Modèle des détecteurs pour 2 positions

La modélisation des détecteurs pour 3 positions est plus complexe car il faut pouvoir tenir compte de toutes les possibilités d'évolution et éviter l'indéterminisme du modèle (Figure 3-9).

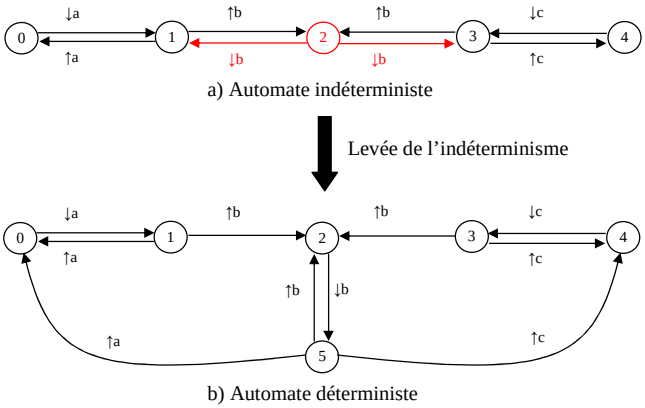


Figure 3-9 : Modèle des détecteurs pour 3 positions

En effet, si le modèle est celui de la Figure 3-9a, alors lorsque le système se trouve sur le détecteur *b* (état 2), il faut être capable soit de continuer jusqu'au détecteur *c*, soit de retourner en *a*. Un indéterminisme est présent lors de la désactivation de l'événement *b*. Cette désactivation engendre un retour soit sur l'état 1, soit sur l'état 3. Le procédé ne peut alors privilégier un état plutôt qu'un autre ce qui conduit à un choix impossible et donc à un indéterminisme. L'indéterminisme est levé par l'état 5 de la Figure 3-9b avec la désactivation

de *b* à partir de l'état 2. Ensuite, ce sont bien deux événements totalement différents qui sortent de l'état 5 pour arriver soit à l'état 0 par l'activation de *a*, soit à l'état 4 par l'activation de *c*.

Il est à noter qu'il n'est pas possible de définir ici une situation initiale du système tant que l'actionneur n'est pas en place. Par la suite, et pour une simplicité de lecture et de compréhension, seuls les systèmes avec 2 détecteurs sont considérés et développés dans ce chapitre.

3.2.3.2. Modèle du préactionneur

Le VDE piloté par un distributeur 5/2 bistable (Figure 3-3) permet de commander le vérin à la fois en sortie par l'ordre *SO* et en rentrée par l'ordre *RE*. La technologie du distributeur associé au vérin engendre un modèle de préactionneur $G_{preac} = (X_{preac}, \Sigma_c, \delta, x_{0preac})$ avec les mêmes spécificités décrites précédemment, c'est à dire :

- Une activation puis désactivation d'ordre ne change pas la position des chambres du distributeur mais implique une réaction du vérin.
- Une priorité sur le premier ordre envoyé existe en cas de conflit de commande.

Reprenons l'exemple du vérin de la Figure 3-3. Il en résulte un modèle à 5 états (Figure 3-10) qui caractérise les spécificités précédentes. De l'état 0, l'ordre *SO* peut être activé (état 1) puis désactivé (retour à l'état 0) et entraîne quand même la sortie du vérin. L'activation de l'ordre *SO* (état 1) puis de l'ordre *RE* (état 3) entraîne une sortie du vérin. De la même manière, si l'ordre *RE* est activé de l'état initial puis *SO*, le vérin continue sa rentrée (état 2 puis état 4). Ces deux situations avec les mêmes ordres envoyés ne répondent pas de la même façon et sont donc bien différentes. De même, à partir de l'état 3, si l'ordre *SO*, qui était prioritaire, est désactivé (état 2), alors c'est l'ordre de rentrée *RE* qui devient prioritaire et fait rentrer le vérin puisqu'il est toujours activé.

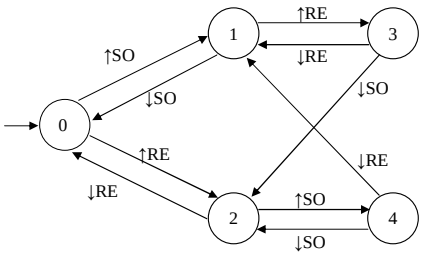


Figure 3-10 : Modèle préactionneur pour le VDE avec distributeur 5/2 bistable

Il est à noter que pour notre étude nous ne souhaitons par diagnostiquer les défauts susceptibles de survenir sur le préactionneur. Seuls les séquences d'événements sur le distributeur nous intéresse et non pas ses situations. Par conséquent, on peut constater que le modèle ci-dessus a la particularité de représenter les deux positions de la chambre à travers le seul état 0 lorsque les deux ordres ne sont pas envoyés. Cet état initial représente donc un indéterminisme du point de vue de la situation physique du distributeur, mais nous verrons qu'il n'en entraîne pas sur le vérin au niveau du modèle final de PO. Une solution pour connaître l'état du distributeur à l'initialisation est de forcer son état en considérant que le vérin doit être en position de rentrée à l'état initial. Le nouvel état initial du modèle préactionneur serait alors l'état 2 où le distributeur est forcément en position chambre droite active.

Si l'on souhaite lever cet indéterminisme afin de considérer les défauts sur le préactionneur, il faut alors prendre en compte chaque situation de la chambre du distributeur. Le modèle résultant est alors un modèle à 6 états où l'état initial nécessite d'être déterminé en fonction de la position du distributeur. Pour de plus ample information sur ce modèle de préactionneur, nous renvoyons le lecteur aux travaux de B. Rohée [ROH 06] concernant une méthodologie de conception et de vérification des modèles de PO.

3.2.3.3. Modèle de l'actionneur

Le modèle de l'actionneur consiste à spécifier les réactions des détecteurs sur chaque état du modèle de préactionneur. A partir de l'état 0, lorsque l'ordre *SO* est envoyé (état 1), le vérin désactive le détecteur *a* pour activer celui de *b*. Cette spécification technologique est confirmée par l'intégration des boucles d'événements non commandables dans le modèle actionneur (Figure 3-11). Il montre les différentes réactions des détecteurs et ceci notamment sur l'état initial où l'on voit les réactions du vérin après activation et désactivation des ordres. De même, sur les états 3 et 4 qui conservent respectivement la priorité sur les réactions aux ordres des états 1 et 2, les boucles d'événements non commandables sont répercutés. Ainsi, l'envoi de l'ordre *SO* puis *RE* (état 3) entraîne bien la sortie du vérin par l'ordre de sortie prioritaire, ce qui engendre la désactivation de *a* puis l'activation de *b*. Il en résulte un modèle actionneur $G_{act} = (X_{act}, \Sigma, \delta, x_{0ac})$.

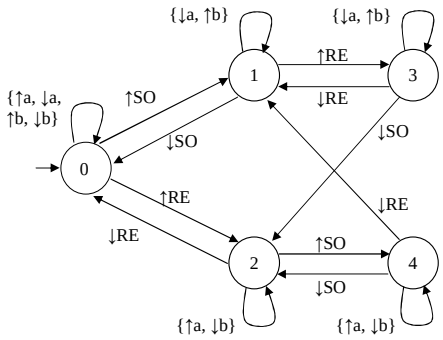


Figure 3-11 : Modèle actionneur pour le VDE avec distributeur 5/2 bistable

La construction du modèle ne peut donc plus se faire de manière totalement automatique mais demande une intervention de l'expert qui va aider à la construction des différents modèles et valider chacun d'entre eux. En effet, l'intégration des boucles d'événements non commandables est un travail d'expert complexe qui demande une connaissance à la fois du modèle et de la technologie de l'EPO concerné.

3.2.3.4. Modèle de Partie Opérative

Au final, le modèle pratique d'EPO_i est obtenu en effectuant la composition synchrone entre le modèle des détecteurs, où un état initial est imposé, et le modèle actionneur. Le modèle d'EPO_i est représenté par un automate $G_i = (X_i, \Sigma_i, \delta_i, x_{0i})$ où $G_i = G_{det} \parallel G_{act}$ et décrit le comportement total du composant à travers un langage $L(G_i)$. Le modèle final du VDE avec un distributeur 5/2 bistable et deux détecteurs est obtenu par la composition du modèle des détecteurs G_{det} de la Figure 3-8 et du modèle actionneur G_{act} de la Figure 3-11. Il en résulte un automate à 15 états et 42 transitions décrivant la totalité de la l'EPO et cela de manière totalement indépendante de la commande (Figure 3-12).

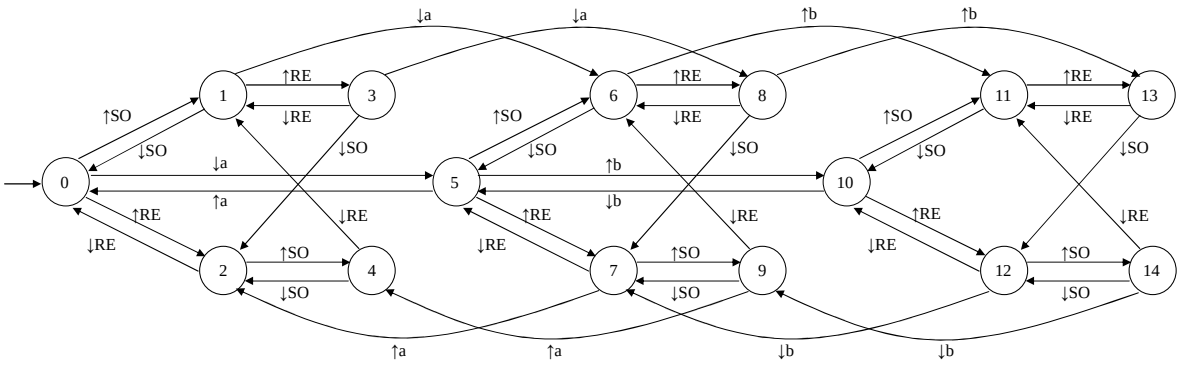


Figure 3-12 : Modèle pratique de la PO du VDE avec distributeur 5/2 bistable

Nous constatons ici que le modèle représente bien les différentes situations physiques du vérin malgré l'indéterminisme de l'état du distributeur. Les états 0, 1, 2, 3 et 4 correspondent au vérin avec le tige rentrée sur le détecteur *a* avec l'ensemble des ordres qu'il est possible d'envoyer de la commande. Les états 5, 6, 7, 8 et 9 décrivent la tige du vérin en position intermédiaire. Enfin, les états 10, 11, 12, 13 et 14 représente la tige du vérin sortie jusqu'au détecteur *b*. Le modèle de PO décrivant l'ensemble des situations physiques du distributeur est donné dans [ROH 06] à travers un automate à 18 états.

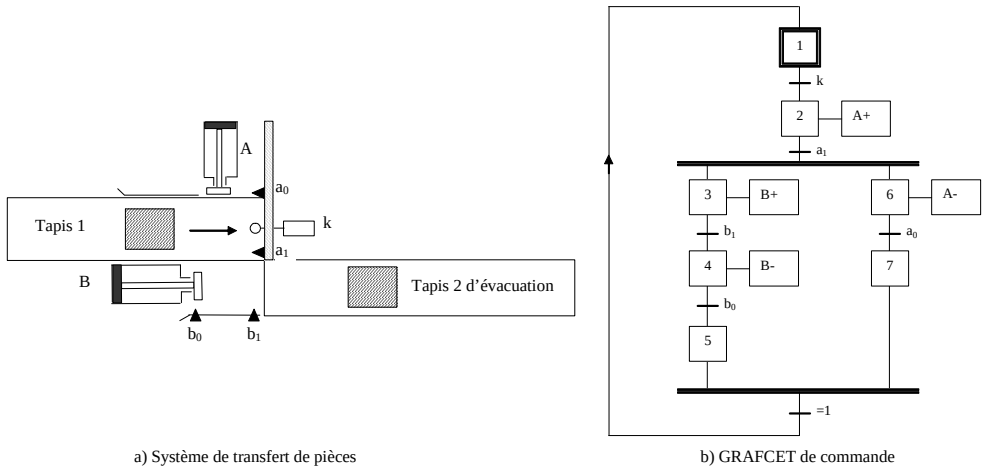
3.3. Intégration de la commande

Après avoir constitué le modèle pratique de chaque composant de la PO d'un procédé, il faut pouvoir y intégrer les informations du cahier des charges à travers un modèle des spécifications. C'est la seconde étape de l'approche de diagnostic décentralisée proposée. La Partie Commande (PC) établit ses spécificités et représente le comportement du fonctionnement normal du procédé. La commande exprime des contraintes de sécurité, ce que l'on ne doit pas faire, et de vivacité, ce que l'on doit faire, sur le procédé. Cependant, nous souhaitons établir un diagnostic décentralisé autour de modèles locaux à partir d'une modélisation de la PO totalement modulaire. La spécification de la commande doit alors être intégrée aux différents modèles obtenus auparavant. Cette intégration conduit à l'obtention des Eléments de Partie Opérative Commandé $EPOC_i$ avec $i \in \{1, 2, \ldots, n\}$. Pour chaque modèle d'$EPOC_i$ décrivant le comportement local désiré, un automate $C_i = (X_{Ci}, \Sigma_i, \delta_i, x_{0i})$ est représenté autour d'un langage de spécification local $K_i = L(C_i)$. Dès lors, il est possible de dire que le langage désiré d'un $EPOC_i$ est inclus dans le langage de comportement possible de l'EPO_i, d'où $K_i \subseteq L(G_i)$.

Certains outils sont mieux adaptés à la modélisation de la PC. Dans un premier temps, nous allons utiliser une spécification de la commande modélisée par GRAFCET. Cet outil, très répandu dans le milieu industriel, constitue une information primordiale sur le comportement désiré du procédé. Une autre manière d'intégrer l'information de la commande est de contraindre chaque modèle de PO jusqu'à son fonctionnement désiré. Cette modélisation de la commande par spécifications de contraintes de sécurité et de contraintes de vivacité est développée en second point.

Pour illustrer nos propos sur l'intégration de la commande, prenons l'exemple de la Figure 3-13a représentant un système de transfert de pièces composé de deux VDE pilotés par des distributeurs 5/2 bistables et de deux moteurs pour les tapis roulants. Ce procédé consiste à

charger des pièces à cadence irrégulière sur l'extrémité gauche du tapis 1. Un automatisme de transfert contrôle le moteur du tapis 1 pour le chargement et le moteur du tapis 2 pour l'évacuation. Ces 2 tapis sont considérés comme toujours en fonctionnement et ne sont donc pas modélisés pour cet exemple d'application. Les vérins A et B sont commandés en sortie par les ordres A+ et B+ ; et en rentrée par les ordres A- et B-. Le procédé dispose d'un capteur k pour la présence des pièces sur le tapis 1. Les capteurs a_0, a_1, b_0 et b_1 sont, quant à eux, associés aux vérins A et B et correspondent aux fins de course de ces vérins.



a) Système de transfert de pièces b) GRAFCET de commande

Figure 3-13 : Commande d'un système de transfert de pièces

3.3.1. Commande spécifiée par GRAFCET

Le GRAFCET de commande est représenté en Figure 3-13b où à partir de l'étape initiale le procédé attend l'arrivée d'une pièce par le capteur k. Le vérin A sort pour pousser la pièce devant le vérin B qui va à son tour sortir pendant que le vérin A retourne à sa position initiale. Lorsque le vérin B a poussé la pièce sur le tapis d'évacuation, il rentre alors en position b_0 par la désactivation de l'ordre B+ et l'occurrence de l'ordre B-. Lorsque les deux vérins sont rentrés, le système peut attendre une nouvelle pièce pour recommencer un nouveau cycle. Pour cet exemple, nous faisons l'hypothèse qu'aucun traitement de diagnostic n'est effectué sur le capteur k puisqu'il représente un événement capteur non influencé par un ordre d'un des deux vérins.

L'objectif de notre étude ne concerne pas la réalisation de la commande, mais bien l'utilisation de celle-ci pour la construction des diagnostiqueurs locaux. Pour cela, la commande spécifiée par GRAFCET est considérée comme "optimale". Elle est obtenue par exemple, par synthèse de commande à travers les travaux de [NDJ 99] et [TAJ 05]. Elle possède donc des propriétés de réactivité, de déterminisme et de sûreté de fonctionnement. Précisons que cette commande "optimale" n'est utilisée que pour l'obtention hors ligne des différents $EPOC_i$ locaux mais n'est en aucun cas la commande implantée en ligne par l'utilisateur. En effet, nous distinguons la commande utilisée hors ligne comme information fonctionnelle pour la construction de nos modèles de la commande implantée sur le système. Le GRAFCET est donc l'outil de modélisation utilisé pour l'intégration de la commande optimale dans nos modèles EPO mais n'impose pas de langage particulier pour la commande implantée en ligne qui pourra, quant à elle, contenir des défauts ou erreurs d'implantation.

L'intégration de la commande s'effectue en procédant à une "intersection" locale entre la

PO et la PC. Pour cela, nous avons développé deux algorithmes d'intersection locale qui conduisent au même résultat mais dans des proportions différentes en terme d'explosion combinatoire.

3.3.1.1. Algorithme 1 d'intersection

L'algorithme 1 d'intersection est basé sur les trois étapes présentées en Figure 3-14 :

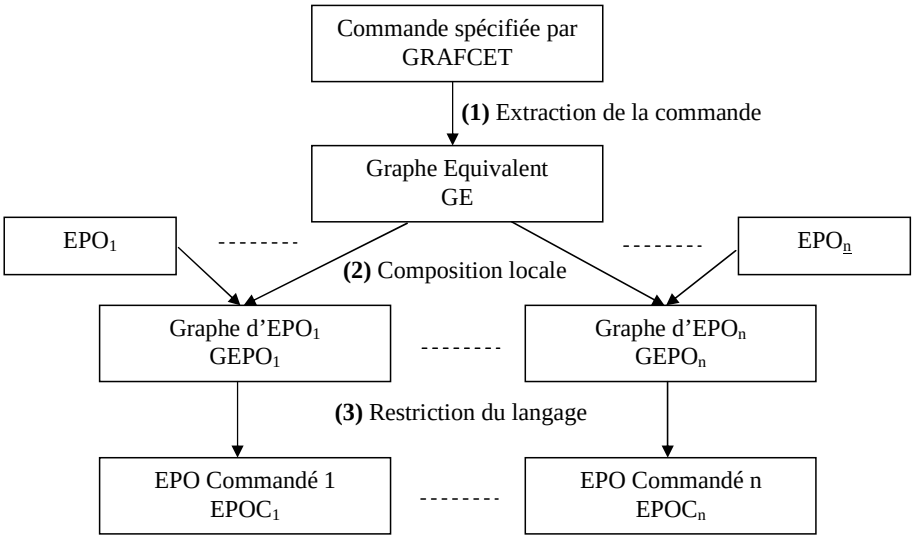


Figure 3-14 : Algorithme 1 d'intersection entre la PO et la PC

La première étape consiste à établir un langage commun entre la PC et la PO. En effet, le GRAFCET possède une sémantique particulière bien différente de celle des automates à états des EPO. Pour cela, nous avons défini un algorithme permettant d'extraire un automate équivalent au GRAFCET appelé Graphe Equivalent *GE*.

La deuxième étape consiste à composer de manière synchrone le Graphe Equivalent *GE* et l'EPO_i concerné. A travers cette étape, le comportement de l'EPO_i est obtenu de manière détaillée. Cependant, la composition comprend également le langage des autres EPO non concernés mais décrits dans le *GE*. Il en résulte un Graphe d'Elément de PO_i ($GEPO_i$).

La troisième et dernière étape concerne la restriction de l'automate $GEPO_i$ au langage de l'EPO_i. Pour cela, il suffit de ne pas considérer le langage des autres EPO en considérant les événements des EPO non concernés comme non observables. C'est une agrégation des états du $GEPO_i$ atteints par les événements n'appartenant pas à l'EPO_i dans des "groupes" d'états. Cette agrégation équivaut à une projection naturelle $P_{L(EPOi)}(GEPO_i)$ du $GEPO_i$ sur le langage de l'EPO_i, $L(EPO_i)$ (Annexe 1). Ensuite, il faut vérifier si tous les groupes sont atteignables et non bloquants. Tout groupe non atteignable ou bloquant doit être supprimé afin de garantir la vivacité du modèle. Au final, un groupe d'états va correspondre à un état de l'automate d'Elément de PO Commandé $EPOC_i$ ne comportant que les événements de l'EPO_i concerné.

a) Génération du Graphe Equivalent

La notion d'automate équivalent au GRAFCET a été vue dans [CHA 99], [KAT 04], [ROU 94] dans le cadre de la synthèse de la commande par GRAFCET. Leur approche consiste à transformer un superviseur GRAFCET en un automate pour l'utilisation de la théorie de supervision selon Ramadge et Wonham [RAM 89a]. L'automate résultant décrit

alors le comportement désiré du procédé à travers des fonctions logiques sur les transitions. Pour nos travaux de diagnostic, nous souhaitons décrire le comportement du procédé le plus précisément possible et cela à l'aide d'un automate événementiel basé sur l'utilisation des fronts montant et descendant. L'obtention du graphe équivalent au GRAFCET est donc particulière dans notre étude puisque nous cherchons à transformer le GRAFCET en automate événementiel où le passage d'un état à un autre est lié à l'apparition d'un événement.

L'algorithme de génération du Graphe Equivalent est obtenu à partir d'un GRAFCET supposant quelques hypothèses qui mériteront d'être levées dans de futurs travaux :

- Le GRAFCET est obtenu après algorithme de synthèse [NDJ 99], [TAJ 05].
- Il ne doit pas comporter de réceptivités temporelles.
- Les actions conditionnelles ne sont pas considérées.

Les détails de l'algorithme 1 d'intersection sont donnés Figure 3-15. Il consiste à créer pour toute étape Et_i d'un GRAFCET d'ordre associée Act et de transition sortante, t_i, un ensemble d'états reliés par des transitions représentant : 1) la désactivation de l'ordre de l'étape précédente, Act_{i-1}, si cet ordre n'est plus activé dans l'étape courante, 2) l'activation de l'ordre associé à l'étape, Act_i, 3) l'intégration de tous les événements non commandables, Σ_{ucActi}, associés à l'ordre, Act_i, de l'étape Et_i par des transitions bouclantes, et 4) le passage à l'étape suivante par soit l'activation de la transition t_i si celle-ci est conditionnée par un passage à 1, soit par sa désactivation si le passage est conditionné par une mise à 0 de t_i.

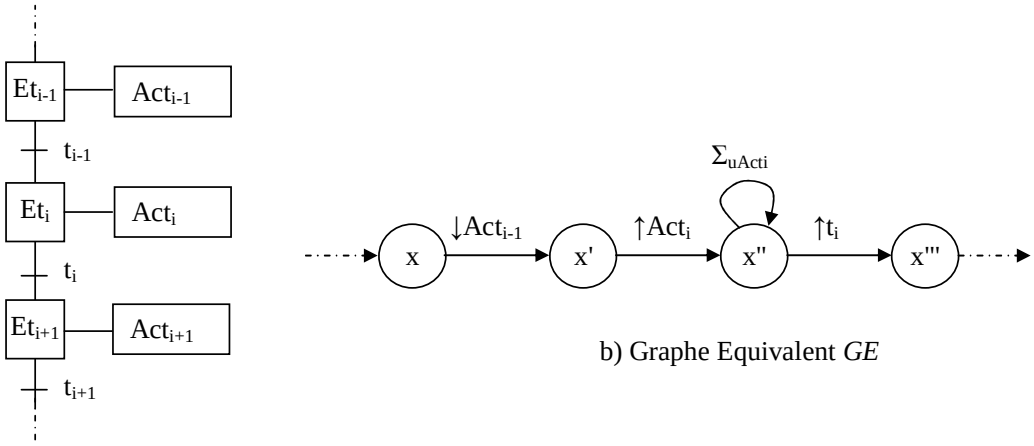
$\forall$ $Et_i \in$ GRAFCET tq $Act_i \in \Sigma_c$ de Et_i et $(Et_i, t_i) = Et_{i+1}$ où $t_i \in \Sigma_{uc}$

Alors $\exists$ x, x', x'', x''' $\in$ *GE* tq

1) $\delta(x, \downarrow Act_{i-1}) = x'$ si $Act_{i-1} \neq Act_i$
2) $\delta(x', \uparrow Act_i) = x''$
3) $\delta(x'', \Sigma_{ucActi}) = x''$
4) $\delta(x'', \uparrow t_i) = x'''$ si $t_i = 1$ ou $\delta(x'', \downarrow t_i) = x'''$ si $t_i = 0$.

Figure 3-15 : Algorithme d'obtention du Graphe Equivalent au GRAFCET

La Figure 3-16 illustre cet algorithme où l'étape Et_i doit désactiver l'ordre, Act_{i-1}, de l'étape précédente. Cet événement correspond au passage de l'état x à l'état x' sur le *GE*. L'envoi de l'ordre Act_i se produit par la suite permettant de passer de l'état x' à l'état x'' de l'automate *GE*. Durant l'exécution de cet ordre, il est possible d'avoir n'importe quel événement non commandable, Σ_{uActi}. Le *GE* exprime cette caractéristique par une boucle d'événements non commandables sur l'état x''. Il faut attendre l'événement non commandable t_i pour sortir de cette étape du GRAFCET et par conséquent quitter l'état x'' du *GE*.



a) GRAFCET

b) Graphe Equivalent *GE*

Figure 3-16 : Passage du GRAFCET au Graphe Equivalent GE

b) Composition synchrone locale

L'étape de composition synchrone entre l'automate *GE* et un EPO_i aboutit à un automate appelé Graphe d'EPO_i (*$GEPO_i$*) qui exprime le comportement désiré de l'EPO_i. Cette étape permet de faire ressortir le comportement désiré détaillé de l'EPO_i, mais elle génère une augmentation du nombre d'états car, dans cette étape, les spécificités des autres modèles d'EPO décrit à travers le *GE*, composent le *$GEPO_i$*. Par conséquent, le graphe enrichi *$GEPO_i$* représente l'automate *GE* de la commande dont on a développé le comportement désiré de l'EPO_i. *GE* exprime le comportement désiré de la commande sans détailler l'ensemble de ces événements non commandables. La composition synchrone de *GE* avec l'EPO_i permet ainsi de développer les événements non commandables de l'EPO_i dans le *GE* et donc de détailler le comportement désiré de l'EPO_i uniquement.

c) Restriction du langage

Pour obtenir uniquement le comportement désiré de l'élément de PO, il faut passer par une étape de restriction du langage qui génère une agrégation des états reliés par des événements n'appartenant pas au langage désiré. C'est une projection naturelle du *$GEPO_i$* sur le langage de l'EPO_i. Pour cela, il faut tout d'abord constituer les différents groupes d'états reliés entre eux par les événements considérés comme non observables pour l'EPO_i concerné.

L'algorithme utilisé pour réaliser le regroupement des états en Groupe est présenté Figure 3-17. A partir d'un état x du *$GEPO_i$* appartenant à un groupe Gr_i, il est possible d'évoluer soit par un événement non observable permettant d'atteindre un état x' appartenant au même groupe, soit par un événement observable impliquant que l'état atteint appartient à un nouveau groupe Gr_{i+1}. Il faut vérifier qu'un même état ne fait pas partie de deux groupes différents. En effet, si un état appartient à deux groupes, cela signifie que les deux groupes peuvent être confondus et regroupés en un seul.

$\forall x \in GEPO_i$ tq $x \in Gr_i$

$\forall \delta(u, x) = x'$ tq $u \in \Sigma_{uoGEPOi}$

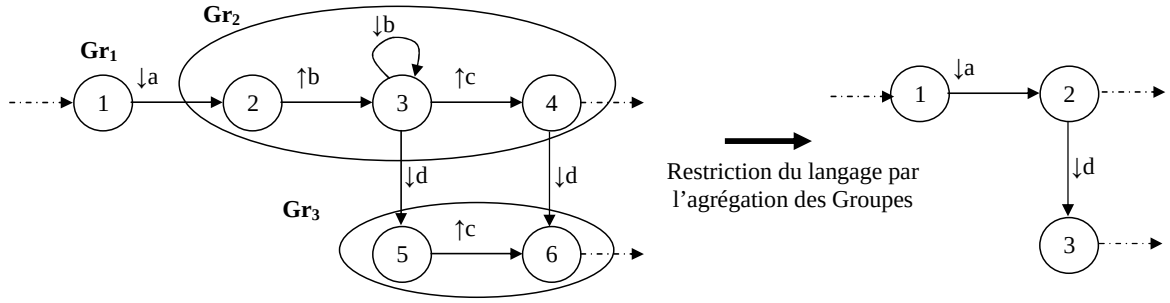
Alors $x' \in Gr_i$

Sinon $u \in \Sigma_{oGEPOi}$ et Alors $x' \notin Gr_i$ mais $x' \in Gr_{i+1}$

Si $x' \in Gr_i$ et $x' \in Gr_{i+1}$ Alors $Gr_i = Gr_i \cup Gr_{i+1}$

Figure 3-17 : Algorithme de regroupement des états

Illustrons nos propos autour de la Figure 3-18 qui présente un automate $GEPO_i$ avec $\Sigma_{GEPOi} = \{a, b, c, d\}$. L'ensemble des événements observés par l'EPO_i est $\Sigma_{EPOi} = \{\uparrow a, \downarrow a, \uparrow d, \downarrow d\}$. Dès lors, l'état 1 dont l'événement sortant est $\downarrow a$ représente un groupe Gr_1 à lui seul. L'événement $\downarrow a$ permet d'arriver à un nouveau groupe Gr_2 qui comprend les états 2, 3 et 4 reliés par les événements $\uparrow b$ et $\uparrow c$ non observés par l'EPO_i. De ce groupe 2, il est possible de sortir par l'événement $\downarrow d$ observable sur l'EPO_i dans un nouveau groupe Gr_3. Il en est de même pour l'ensemble des autres états de l'automate. L'agrégation des états dans un groupe correspond à un état de l'automate représentant le comportement désiré de l'EPO_i. Cet automate constitue un EPO Commandé $EPOC_i$.



a) Automate $GEPO_i$ b) Automate $EPOC_i$

Figure 3-18 : Restriction du langage

Le principe de restriction du langage par groupe est très proche de la théorie des régions développée pour la synthèse de la commande dans [NDJ 99], [TAJ 05]. Cependant, il faut bien voir ici une restriction du langage désiré et non pas un regroupement des événements commandables représentant les états stables d'un procédé.

3.3.1.2. Application de l'algorithme 1 d'intersection sur l'exemple

a) Génération de l'automate de Graphe Equivalent *GE*

La première étape d'intégration de la commande dans les EPO consiste à réaliser l'automate *GE* du GRAFCET du système de transfert de pièces (Figure 3-19). A partir de l'état 1, l'événement attendu est ↑k qui permet d'arriver à l'état 2 correspondant à l'étape 2 du GRAFCET. Dans cette étape, l'ordre A+ est envoyé, ce qui correspond au passage de l'état 2 à l'état 3 dans le *GE*. Dès lors, tous les événements associés au vérin A peuvent survenir. Cette caractéristique est représentée par la boucle d'événements sur l'état 3. Il faut attendre l'événement $\uparrow a_1$ pour passer à l'état 4 dans le *GE* correspondant aux étapes 3 et 6 du GRAFCET où l'ordre A+ doit être désactivé (état 5).

La séquence simultanée du GRAFCET est représentée par une structure parallèle sur le *GE*. La branche gauche du GRAFCET composée des étapes 3, 4 et 5 est représentée par la structure verticale dans le *GE* où l'activation de l'ordre B+ permet d'atteindre l'état 9 avec les événements appartenant au vérin B rebouclant sur cet état, puis les états 13, 17, 21, 25 puis 29. La branche droite du GRAFCET (étapes 6 et 7) est représentée par la structure horizontale du *GE* où l'activation de l'ordre A- (état 6) permet la rentrée du vérin A jusqu'en a_0 (état 7) avant d'être désactivé (état 8). Les états 10, 11, 12, 14, 15, 16, 18, 19, 20, 22, 23, 24, 26, 27, 28, 30, 31 et 32 décrivent l'ensemble des possibilités d'évolution des événements sur la structure simultanée du GRAFCET. L'état 32 du *GE* correspond donc à la situation 5 et 7 du GRAFCET où la synchronisation du GRAFCET permet le retour en étape initiale par la transition dont la réceptivité est toujours vraie. Ceci engendre pour le *GE* le retour en état initial.

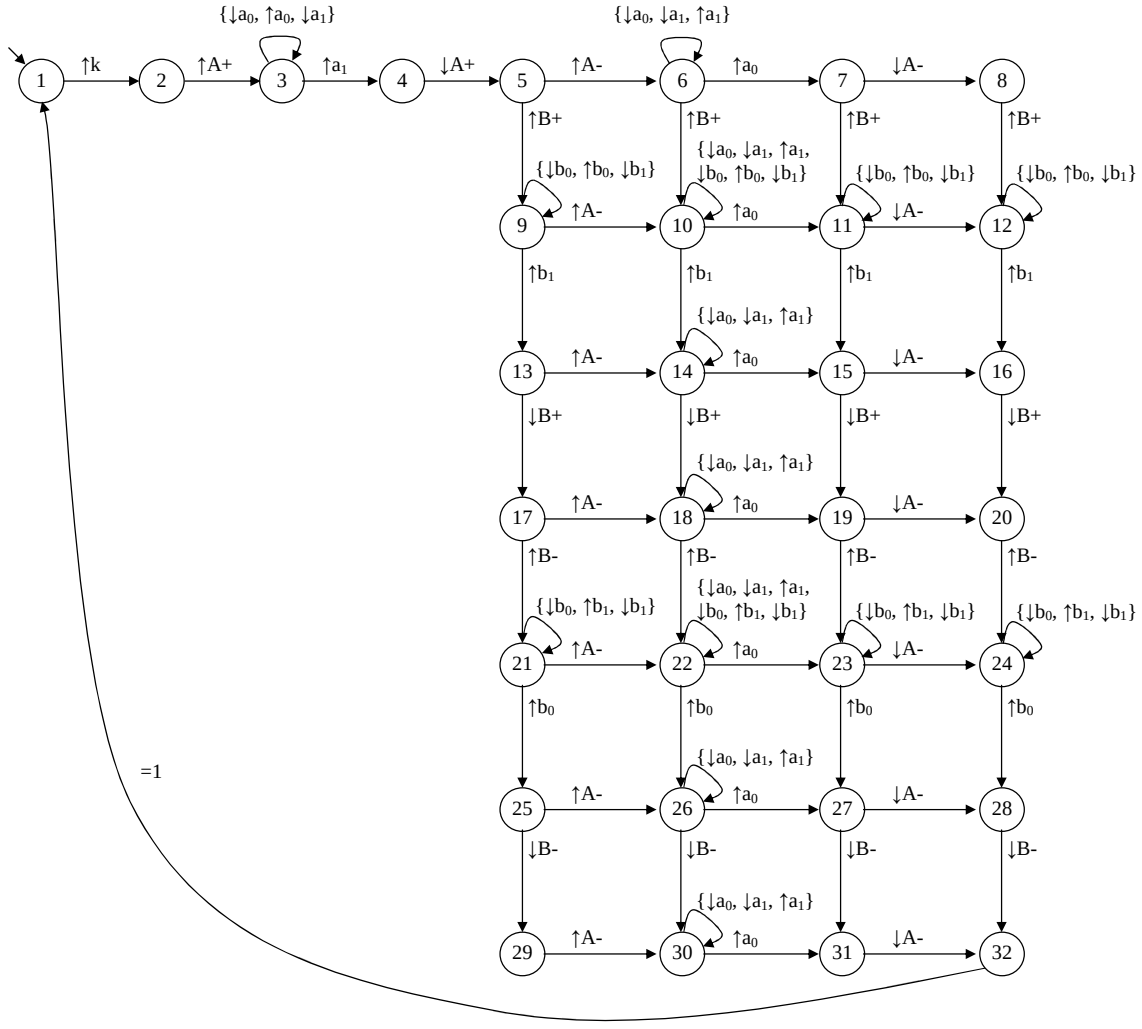
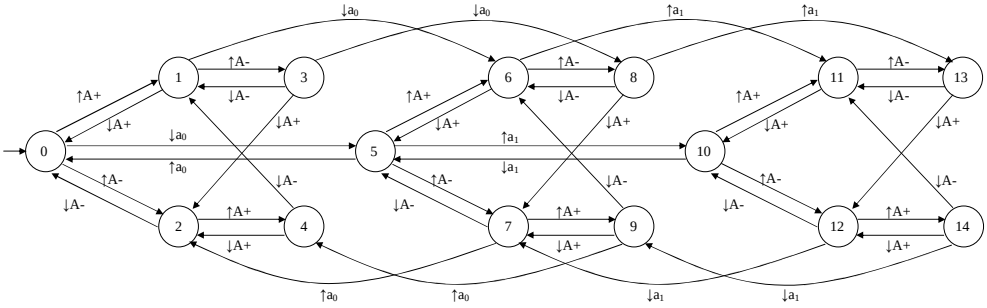


Figure 3-19 : Graphe Equivalent GE du GRAFCET du transfert de pièces de la Figure 3-13

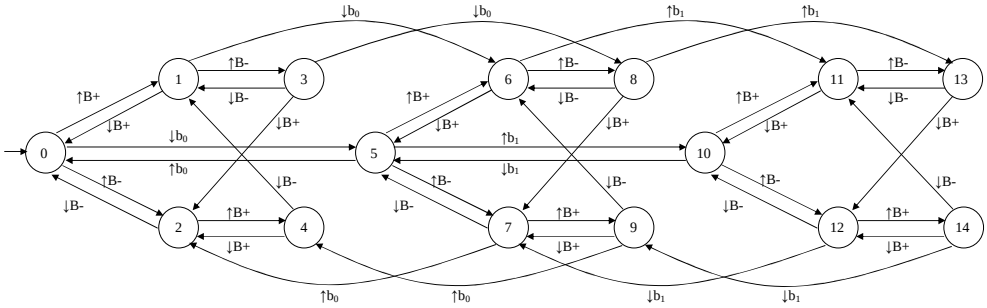
D'un GRAFCET à 7 étapes et 6 transitions, on élabore un Graphe Equivalent *GE* composé de 32 états et 64 transitions dont 14 boucles. Sur cet exemple, il n'est pas possible de parler d'explosion du nombre des états puisque c'est essentiellement la structure parallèle donnée au GRAFCET qui rend le Graphe Equivalent plus ou moins important. Nous verrons, dans le chapitre 4, différentes applications nous permettant de discuter de cet aspect lié à la complexité.

b) Composition synchrone locale

Après avoir obtenu l'information de la commande sous forme d'automate, il faut l'intégrer localement sur chaque EPO du système. Afin de réaliser l'intersection entre la commande et les EPO, nous avons besoin de choisir les modèles correspondant aux éléments de la PO du procédé. Pour l'exemple de la Figure 3-13, ce sont deux VDE pilotés par des distributeurs 5/2 bistables avec deux détecteurs de fin de course. Leur modèle est représenté en Figure 3-20. Ils comportent chacun 15 états parmi lesquels il existe le comportement désiré que la commande doit parcourir.



a) Modèle de PO du vérin A : EPO_A



b) Modèle de PO du vérin B : EPO_B

Figure 3-20 : Modèles pratiques des EPO des vérins A et B de l'exemple de la Figure 3-13

Il faut effectuer la composition synchrone entre le *GE* avec l'EPO_A du vérin A puis avec l'EPO_B du vérin B. Le résultat de la composition synchrone du *GE* du système de transfert de pièces et de l'EPO du vérin A est représenté en Figure 3-21. C'est un Graphe d'EPO_A du vérin A : $GEPO_A = GE \parallel EPO_A$, composé de 40 états intégrant les réactions particulières exprimées par l'EPO_A du vérin A. On peut voir par exemple que la désactivation de a_0 (état 4 de la Figure 3-21) est une conséquence de l'ordre A+. Ce sont les événements non commandables du vérin A qui étaient présents auparavant dans les boucles de l'automate *GE* qui sont détaillés par la création d'états caractéristiques décrivant le comportement désiré de l'EPO_A dans le *GE*. Pour une question de lisibilité, nous n'avons pas représenté la composition synchrone du *GE* avec l'EPO_B du vérin B mais il en est bien sûr du même principe.

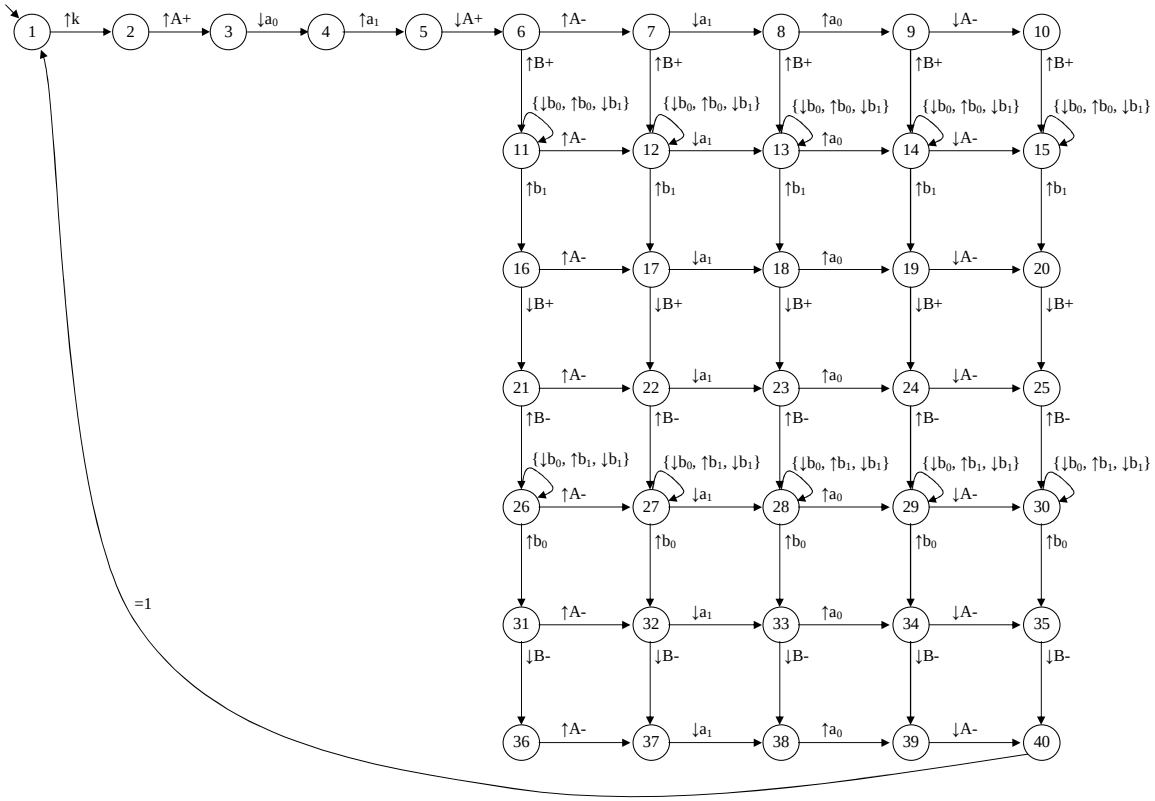


Figure 3-21 : Composition synchrone GE || EPO_A = $GEPO_A$

c) Restriction du langage

La dernière étape est la restriction du langage. Pour le vérin A, il faut partir de l'automate $GEPO_A$ et définir les groupes représentant le langage désiré de l'EPO_A du vérin A (Figure 3-22). Les événements concernés sont $\Sigma_{EPOA} = \{\downarrow a_0, \uparrow a_0, \downarrow a_1, \uparrow a_1, \downarrow A+, \uparrow A+, \downarrow A-, \uparrow A-\}$.

- Les états 1 et 2 sont rassemblés dans un groupe Gr_1 car l'événement k n'appartient pas au langage L_A de l'EPO_A du vérin A.
- Chacun des états 3, 4, 5 constitue respectivement un groupe Gr_2, Gr_3 et Gr_4.
- Les états 6, 11, 16, 21, 26, 31 et 36 sont reliés entre eux par des événements n'appartenant pas au langage L_A de l'EPO_A et constituent donc un groupe Gr_5.
- Le groupe Gr_6 regroupant les états 7, 12, 17, 22, 27, 32 et 37.
- Le groupe Gr_7 constitué des états 8, 13, 18, 23, 28, 33 et 38.
- Le groupe Gr_8 (états 9, 14, 19, 24, 29, 34 et 39).

On remarque également que le groupe Gr_9 rassemble les états 10, 15, 20, 25, 30, 35, 40 mais aussi les états 1 et 2 en raison de la présence de la transition dont la réceptivité est toujours vraie (Figure 3-23a). Dès lors, le groupe Gr_1 est compris dans le groupe Gr_9. Ces deux groupes sont donc communs et ne seront représentés par un seul et même état dans l'automate final $EPOC_A$ obtenu par agrégation (Figure 3-23b).

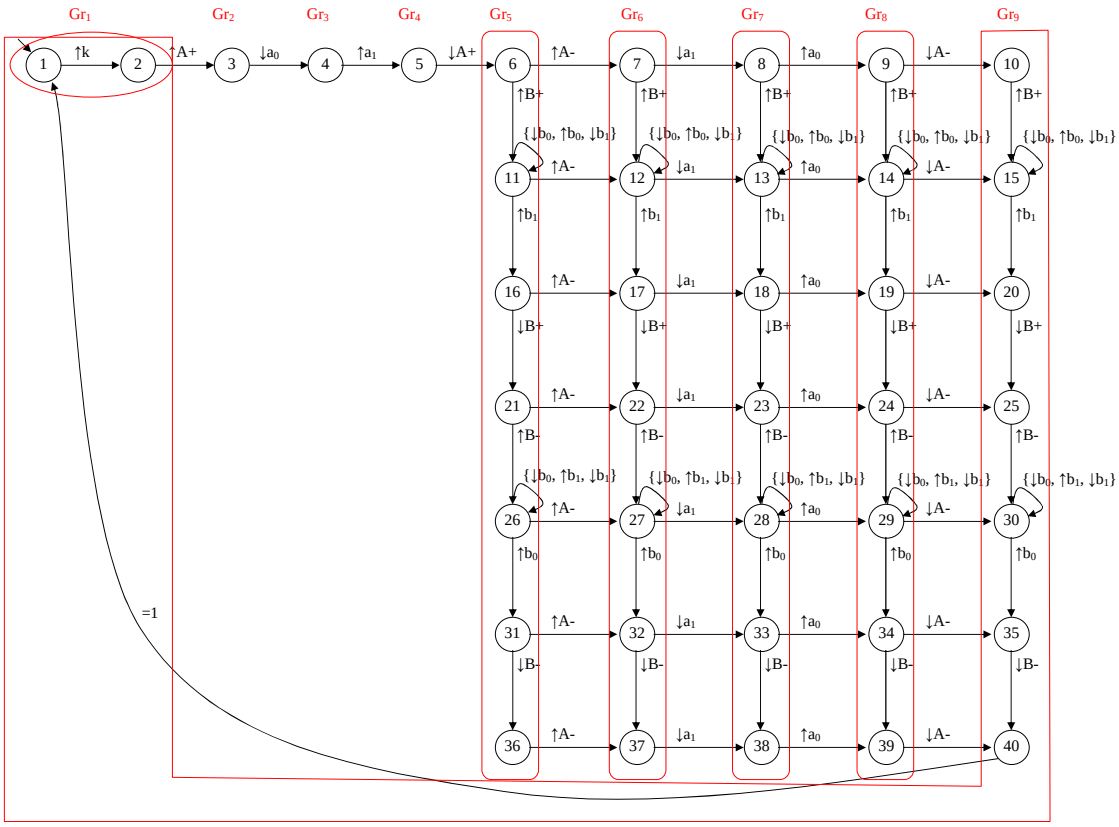
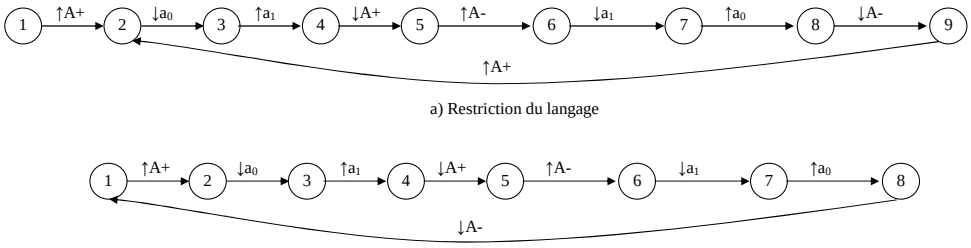


Figure 3-22 : Constitution des groupes sur le $GEPO_A$

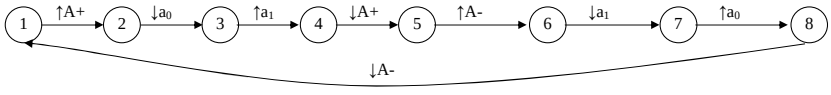


a) Restriction du langage

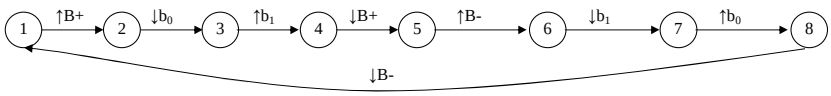
b) Agrégation des groupes communs $EPOC_A$

Figure 3-23 : Obtention de l'automate $EPOC_A$ du vérin A

Il résulte de l'agrégation des groupes, les automates de la Figure 3-24. L'intégration de l'information de la commande permet de modéliser localement la commande à un niveau de description très bas par des EPOC.



a) Modèle EPO Commandé du vérin A : $EPOC_A$



b) Modèle EPO Commandé du vérin B : $EPOC_B$

Figure 3-24 : EPOC du vérin A et B pour le transfert de pièces de la Figure 3-13

3.3.1.3. Algorithme 2 d'intersection

L'étape 2 de l'algorithme 1 d'intersection réalise une composition synchrone entre le modèle complet du Graphe Equivalent *GE* de la commande avec chaque EPO_i. Cette étape introduit un risque d'explosion du nombre d'états avant le passage de la restriction du langage préconisé dans l'étape 3. Afin d'éviter ce risque, nous proposons un second algorithme qui consiste à restreindre le langage de l'automate *GE* au langage désiré de l'EPO_i concerné avant de réaliser la composition synchrone avec l'EPO_i. L'algorithme 2 d'intersection se compose alors de trois étapes illustrées Figure 3-25 :

1) La première étape correspond à la première étape de l'algorithme 1 précédent pour obtenir le Graphe Equivalent *GE* du GRAFCET. L'objectif est de disposer d'une sémantique similaire entre les différents EPO et la commande.

2) La deuxième étape consiste à restreindre le Graphe Equivalent *GE* au langage de l'EPO_i. Il s'agit d'une agrégation des états dans des "groupes" d'états. Ces états étant atteints par des événements non observés par l'EPO_i. Cette agrégation est une projection naturelle $P_{L(EPOi)}(GE)$ du *GE* sur le langage de l'EPO_i, $L(EPO_i)$. Tout groupe non atteignable ou bloquant est supprimé afin de garantir la vivacité du modèle. La restriction du langage permet d'obtenir un *GE* Restreint au langage de l'EPO_i noté $GEREPO_i$. Cette restriction permet ainsi de diminuer le nombre d'états avant la composition synchrone.

3) La troisième étape est la composition synchrone locale entre $GEREPO_i$ et l'EPO_i correspondant. A ce niveau, la composition est réalisée entre deux automates de taille acceptable. Le résultat de cette composition est équivalent au résultat final de l'algorithme 1 d'intersection, c'est-à-dire, un automate décrivant le comportement désiré de l'EPO_i appelé $EPOC_i$.

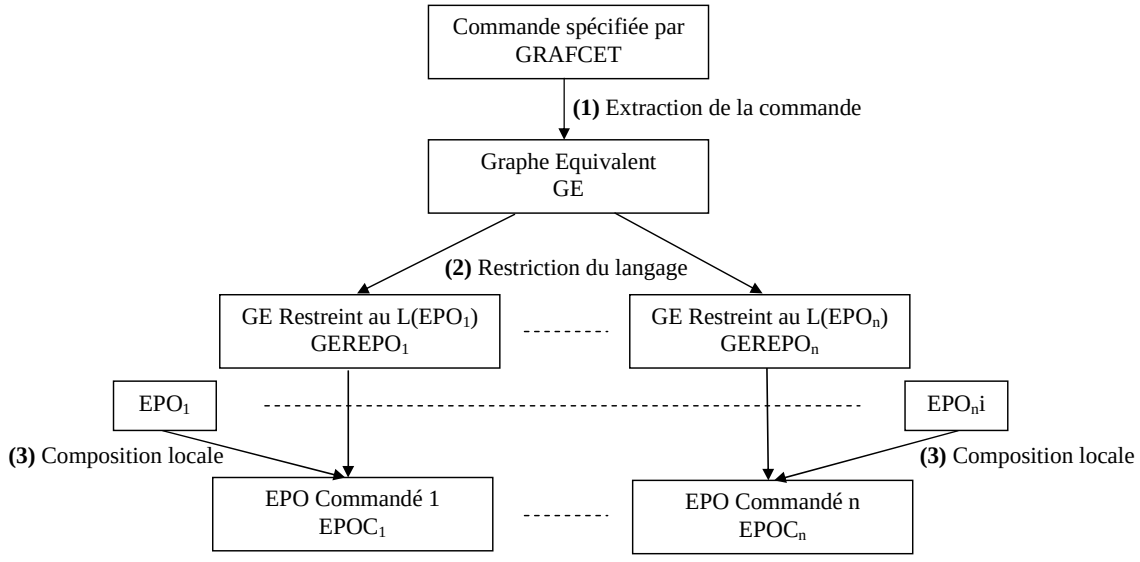


Figure 3-25 : Algorithme 2 d'intersection entre la PO et la PC

3.3.1.4. Application de l'algorithme 2 d'intersection sur l'exemple

Reprenons l'exemple du transfert de pièces de la Figure 3-13. Le Graphe Equivalent GE du GRAFCET est identique à celui proposé Figure 3-19. La deuxième étape de l'algorithme 2 d'intersection consiste à réduire l'automate *GE* au langage de l'EPO_A du vérin A et de l'EPO_B

du vérin B. Pour le vérin A, cette restriction équivaut au regroupement représenté en Figure 3-26. Ainsi, les états 1, 2, 8, 12, 16, 20, 24, 28 et 32 sont regroupés dans le groupe Gr_7 qui inclut le groupe Gr_1 constitué des états 1 et 2. Ce groupe est composé des états qui sont reliés entre eux par un événement non observé par l'EPO_A. L'état 3 du *GE* est un groupe à lui (Gr_2), tout comme l'état 4 (Gr_3). Par contre, les états 5, 9, 13, 17, 21, 25 et 29 sont reliés par des événements appartenant au langage de l'EPO_B et constituent un nouveau groupe Gr_4. Il en est de même pour le groupe Gr_5 composé des états 6, 10, 14, 18, 22, 26 et 30 ; et pour le groupe Gr_6 composé des états 7, 11, 15, 19, 23, 27 et 31. Au final, l'étape de restriction de l'algorithme 2 d'intersection fournit un *GE* restreint au langage de l'EPO_A ($GEREPO_A$) à 6 états (Figure 3-27).

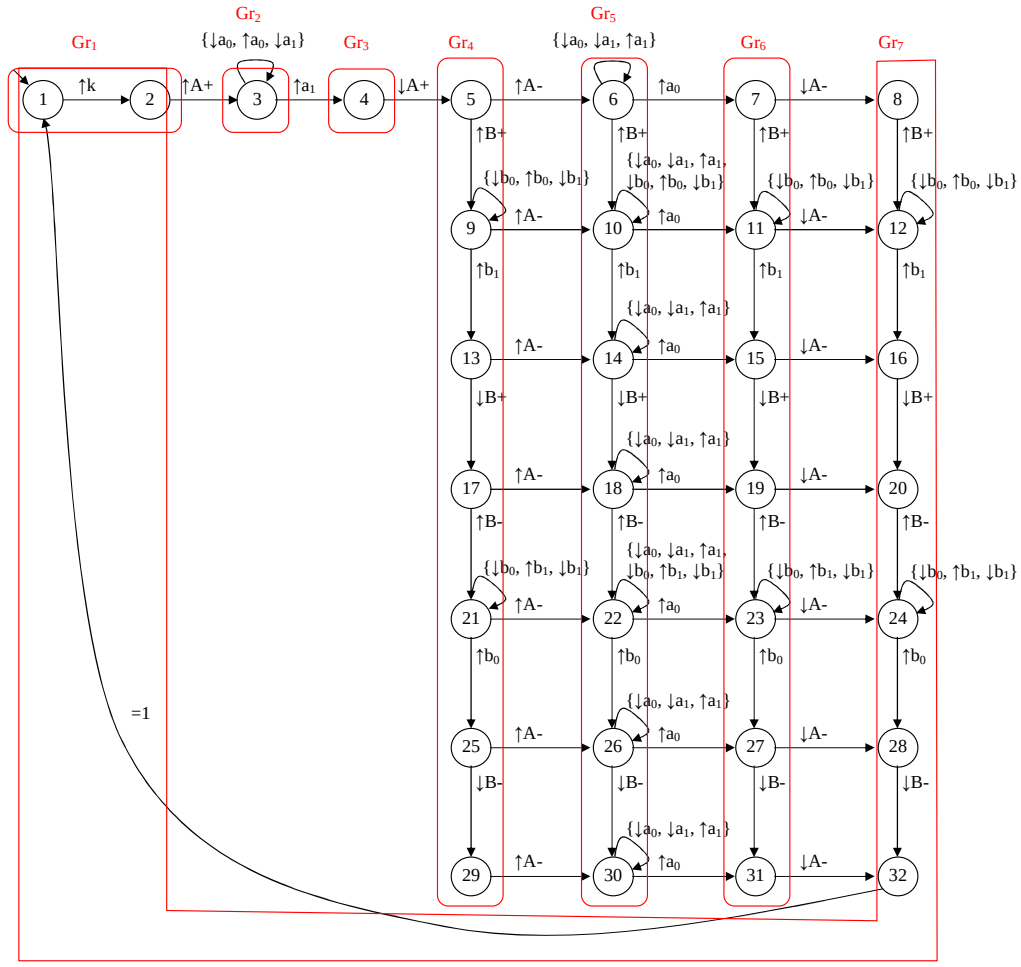


Figure 3-26 : Regroupement des états du GE au langage de l'EPO_A du vérin A

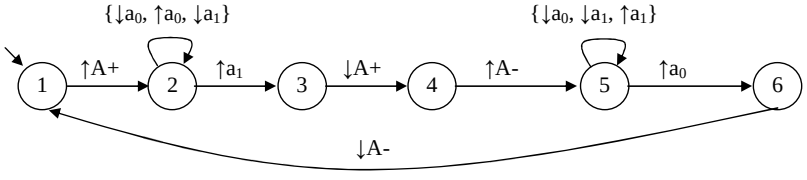


Figure 3-27 : GE Restreint au langage de l'EPO_A : $GEREPO_A$

La troisième étape consiste à réaliser une composition synchrone locale entre l'automate GEREPO de chaque vérin et l'automate de chaque vérin $EPOC_A$ et $EPOC_B$. Ainsi, la composition synchrone entre le graphe $GEREPO_A$ de la Figure 3-27 et l'automate EPO_A de la

Figure 3-20a fournit un automate décrivant le comportement désiré du vérin A en fonction de la commande par l'automate $EPOC_A = GEREPO_A \parallel EPO_A$ (Figure 3-24a). Il en est de même pour l'obtention de l'automate $EPOC_B = GEREPO_B \parallel EPO_B$ de la Figure 3-24b.

3.3.1.5. Conclusion sur les deux algorithmes d'intersection

L'intérêt du second algorithme est qu'il permet d'éviter le passage par une étape où le risque d'explosion combinatoire n'est pas négligeable. Le Tableau 3-2 présente, pour l'exemple du transfert de pièces de la Figure 3-13, un gain important entre les deux algorithmes d'intersection au niveau de l'étape 2 tout en gardant les mêmes performances au final. En effet, pour l'algorithme 1 présente une étape de composition de 40 états alors que l'algorithme 2 n'implique que 6 états dans son étape 2 lors de la restriction de langage. Il apparaît alors clairement que l'algorithme 2 d'intersection est plus intéressant en terme de traitement d'automates. Par la suite, nous n'utilisons que l'algorithme 2 d'intersection. Nous verrons, dans le chapitre 4, son utilisation autour d'un exemple d'application manufacturière.

	Algorithme 1		Algorithme 2	
Etape 1	Extraction : GE	32 états	Extraction : GE	32 états
Etape 2	Composition : $GEPO_A$	**40 états**	Restriction : $GEREPO_A$	**6 états**
Etape 3	Restriction : $EPOC_A$	8 états	Composition : $EPOC_A$	8 états

Tableau 3-2 : Comparaison des algorithmes 1 et 2 d'intersection entre la commande et l'EPO_A du vérin A de la Figure 3-13

3.3.2. Commande spécifiée par des contraintes locales

Nous venons de voir une méthode d'intégration de la commande utilisant un modèle GRAFCET. Cependant, cette solution oblige à passer par une étape de constitution d'un modèle global de la commande. Nous proposons d'étudier une seconde solution en utilisant, pour spécifier la commande, un concept de contraintes de sécurité et de vivacité.

Intégrer des contraintes consiste à inhiber des actions et/ou à agencer et séquencer l'exécution des commandes envoyées au procédé. Une contrainte ne peut pas provoquer d'actions supplémentaires dans un modèle mais peut exprimer une restriction, ou inhibition, de ces actions. La modélisation de ces contraintes peut s'effectuer soit par automates soit par des équations logiques. Les contraintes peuvent s'appliquer soit globalement sur l'ensemble du procédé, soit localement sur chaque EPO_i. Notre démarche reposant sur l'obtention d'une structure décentralisée, nous appliquons uniquement des contraintes locales sur les EPO.

3.3.2.1. Modélisation des spécifications par automates

La modélisation par automates implique de définir les réactions particulières sur les événements. Ces réactions sont parfois difficilement modélisables et cela notamment pour les contraintes de vivacité. Dans [PHI 04a] et [PHI 04b], une méthode de modélisation structurée a été développée. C'est une adaptation, pour les contraintes, de la méthode de modélisation théorique de la partie opérative abordée au paragraphe 3.2.2 de ce chapitre.

La méthode est basée sur les règles d'occurrence et des relations de précédence. La modélisation des contraintes impose de modifier quelque peu les définitions des règles et relations. Dès lors, les règles d'occurrence définissent un lien entre les événements commandables et/ou les événements non commandables qu'ils soient cause ou conséquence. Les relations de précédence définissent quant à elles un lien soit entre événements commandables entre eux, soit entre événements non commandables entre eux.

a) Contraintes de sécurité

Nous définissons les contraintes de sécurité comme étant des contraintes entre les événements commandables et les contraintes de vivacité comme des contraintes entre les événements commandables et les événements non commandables. Par conséquent, la méthode de modélisation des contraintes va dépendre du type de contrainte à modéliser.

La modélisation des contraintes de sécurité consiste à établir tout d'abord la situation initiale de l'élément de PO, puis à définir les relations de précédence utilisées entre les événements commandables. Aucun événement non commandable n'intervenant dans cette contrainte, il n'y a donc aucune règle d'occurrence entre les événements commandables et les événements non commandables.

La construction de la contrainte se fait suivant les étapes suivantes :

- Construction de l'automate à 2^n états avec boucles d'autorisation de tous les événements Σ où n représente le nombre d'ordres.
- Intégrations des règles d'occurrence et relations de précédence
- Suppression des états non atteignables

Prenons l'exemple du transfert de pièces de la Figure 3-13 où une contrainte de sécurité sur le vérin A est de ne pas envoyer l'ordre de sortie en même temps que l'ordre de rentrée. La condition initiale pour modéliser cette contrainte, est de n'avoir aucun ordre activé : A+ = 0 et A- = 0. Il faut établir ensuite les relations de précédence entre les événements de ces deux ordres, à savoir :

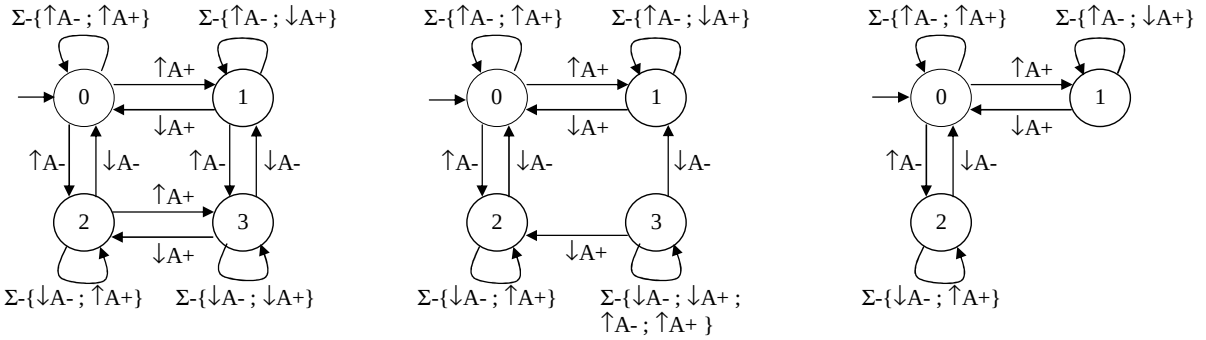
Règles	Relation sur les contraintes
(1) $\uparrow A+ \Rightarrow \Sigma-\{\uparrow A-\}$	$\uparrow A+$ précède tous les événements sauf $\uparrow A-$
(2) $\uparrow A- \Rightarrow \Sigma-\{\uparrow A+\}$	$\uparrow A-$ précède tous les événements sauf $\uparrow A+$
(3) $\downarrow A+ \Rightarrow \Sigma$	$\downarrow A+$ précède tous les événements
(4) $\downarrow A- \Rightarrow \Sigma$	$\downarrow A-$ précède tous les événements

Tableau 3-3 : Relation sur la contrainte "Ne pas Sortir et Rentrer le vérin A en même temps"

Dès lors, la contrainte du vérin A est établie par l'automate de la Figure 3-28a qui représente le comportement le plus permissif possible. A partir des conditions initiales (état 0), il est possible d'envoyer soit l'ordre A+ pour aller vers l'état 1, soit l'ordre A- pour aller vers l'état 2. A partir des états 1 et 2, rien n'interdit d'envoyer les ordres complémentaires pour se retrouver dans l'état 3.

De cet automate, il faut appliquer les relations du Tableau 3-3. Ainsi, à partir de l'état 1, après l'activation $\uparrow A+$, il faut interdire l'événement $\uparrow A-$ qui permettait d'aller à l'état 3. Il en est de même pour l'état 2 où l'on interdit l'événement $\uparrow A+$ (Figure 3-28b). A partir de cet

automate, il faut regarder tous les états afin de supprimer les états qui ne sont plus accessibles ainsi que leurs transitions. Le résultat de cette suppression permet d'obtenir une modélisation de la contrainte à appliquer au modèle de PO (Figure 3-28c).



a) Automate à 2^n états b) Intégration des règles et relations c) Suppression des états non accessibles

Figure 3-28 : Modèle de la contrainte " Ne pas Sortir et Rentrer le vérin A en même temps"

b) Contraintes de vivacité

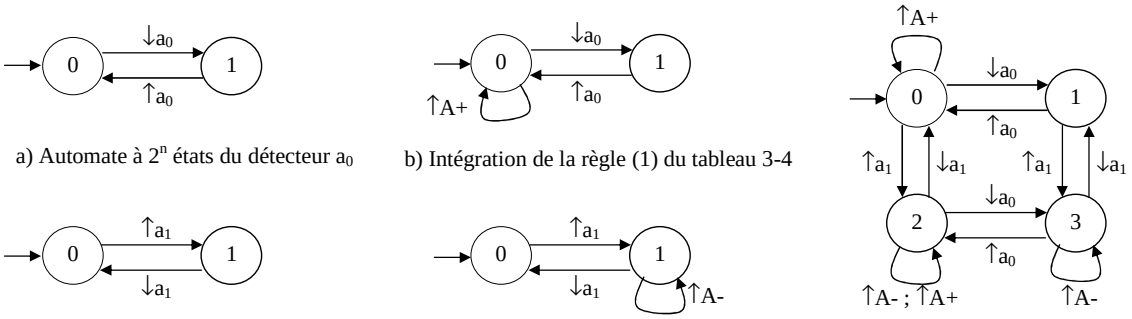
La modélisation des contraintes de vivacité est basée sur le même principe à la seule différence que les règles sont établies entre les événements commandables et les événements non commandables. Chaque règle va ensuite définir un automate de restriction sur le séquencement des événements et le modèle final est alors obtenu par une composition synchrone des automates de règles.

Toujours dans l'exemple du vérin A du système de transfert de pièces (Figure 3-13), il est possible de définir une contrainte de vivacité tel que : "L'activation d'un ordre ne peut se réaliser que si le vérin est en fin de course". En d'autres termes, cela signifie :

Règles	Relation sur les contraintes
(1) $a_0 = 1 \Rightarrow \uparrow A+$	Activation de A+ que si vérin en a_0
(2) $a_1 = 1 \Rightarrow \uparrow A-$	Activation de A- que si vérin en a_1

Tableau 3-4 : Relation sur la contrainte "Activation d'un ordre si vérin A en fin de course"

Nous avons deux règles donc deux automates pour les représenter à partir des conditions initiales que sont $a_0 = 1$ et $a_1 = 0$ (Figure 3-29a). L'intégration des règles dans ces automates introduit des boucles d'autorisation d'événements dans les états respectant ces règles (Figure 3-29b). Enfin, une composition synchrone entre les deux automates de règles modélise la contrainte souhaitée (Figure 3-29c). On remarque que cette contrainte n'exprime pas le fait qu'il est impossible d'avoir les deux détecteurs à 1. C'est l'ensemble des contraintes appliquées aux modèles qui permet d'obtenir le comportement désiré de chaque EPO.



a) Automate à 2^n états du détecteur a_0 b) Intégration de la règle (1) du tableau 3-4

a) Automate à 2^n états du détecteur a_0 b) Intégration de la règle (2) du tableau 3-4 c) Synchronisation des automates de règles

Figure 3-29 : Modèle de la contrainte "Activation d'un ordre si vérin A en fin de course"

L'application de ces contraintes de sécurité et de vivacité sur les modèles de PO s'effectue par composition synchrone de tous les automates. Cette composition d'automates de contraintes est équivalente à un superviseur local défini selon Kumar dans [KUM 91] et retrouvé dans de nombreux travaux sur la synthèse de la commande comme dans [NDJ 99] ou [TAJ 05]. Au final, si aucune contrainte n'a été oubliée, le résultat doit être identique à celui obtenu par la Figure 3-24. Cette modélisation des contraintes par automates nécessite le passage par des compositions synchrones qui sont parfois lourdes d'utilisation. De plus, la construction des automates, bien qu'explicative, reste néanmoins une étape contraignante.

3.3.2.2. Modélisation des spécifications par équations logiques

a) Principe

Pour pallier la lourdeur des processus de modélisation des spécifications pas automates, nous préconisons, pour la modélisation des contraintes, l'utilisation d'équations logiques. La modélisation des contraintes par équations logiques a fait l'objet de divers travaux dans le cadre de la synthèse de la commande [ROU 03], [TAJ 05], [WAN 00]. Ces équations doivent être vérifiées pour chaque état et chaque transition de chaque automate de PO. Les états sont modélisés alors par un vecteur d'état composé de la valeur booléenne de chaque détecteur et de chaque ordre de l'EPO_i concerné. Les équations logiques ont la faculté de pouvoir s'appliquer localement sans passer par une étape de composition. Le cadre formel de la modélisation des contraintes est basé sur l'algèbre de Boole pour laquelle les éléments manipulés sont des valeurs binaires 0 ou 1. Ces éléments sont représentés par des fonctions définies dans IB^{λ} où λ est le nombre de variables d'états, les variables représentant les ordres et les sorties des capteurs. Pour composer les éléments de *IB* entre eux, trois lois de base sont définies :

- La loi ET : $(f, g) \rightarrow f \wedge g$ définie dans $IB^2 \rightarrow IB$
- La loi OU : $(f, g) \rightarrow f \vee g$ définie dans $IB^2 \rightarrow IB$
- La loi NON : $f \rightarrow \neg f$ définie dans $IB \rightarrow IB$

Chaque état et/ou chaque transition du modèle de PO doit vérifier les équations logiques des contraintes. Le non-respect d'une de ces équations entraîne alors la suppression de l'état ou de la transition ne satisfaisant pas la contrainte.

b) Application à l'exemple

Pour l'exemple du vérin A du système de transfert de pièces (Figure 3-13), nous imposons des contraintes de sécurité et de vivacité, liées au déplacement. La première contrainte consiste à interdire l'activation simultanée des ordres A+ et A-. L'équation logique représentant cette contrainte est donnée par : $A+ \wedge A- = 0$. Dès lors, le modèle de la PO du vérin A (Figure 3-20a) interdit d'avoir A+ = 1 et A- = 1 en même temps, ce qui supprime les états 3, 4, 8, 9, 13 et 14 avec leurs transitions sortantes et entrantes pour arriver à l'automate de la Figure 3-30.

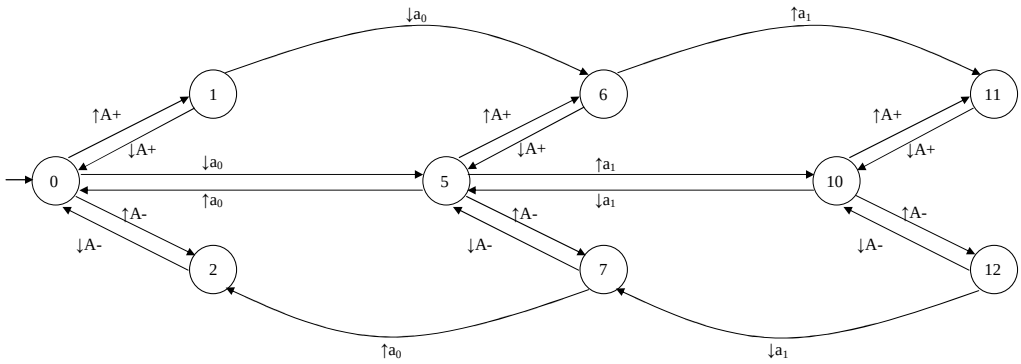


Figure 3-30 : Intégration de la contrainte de sécurité interdisant les ordres A+ et A- simultanés à l'EPO_A

Une autre contrainte de sécurité est de ne pas autoriser l'arrêt du vérin lorsque celui-ci n'est pas dans une position de fin de course. Cette contrainte s'exprime par la condition suivante : $\neg a_0 \wedge \neg a_1 \wedge \neg A+ \wedge \neg A- = 0$. Ainsi, la situation où aucun ordre est activé entre les deux fins de course ne doit pas se produire. Dès lors, l'état 5 du modèle de PO est interdit puisque son vecteur d'état (a_0, a_1, A+, A-) = (0, 0, 0, 0) est interdit par la contrainte. Les transitions associées à cet état sont également supprimées (Figure 3-31).

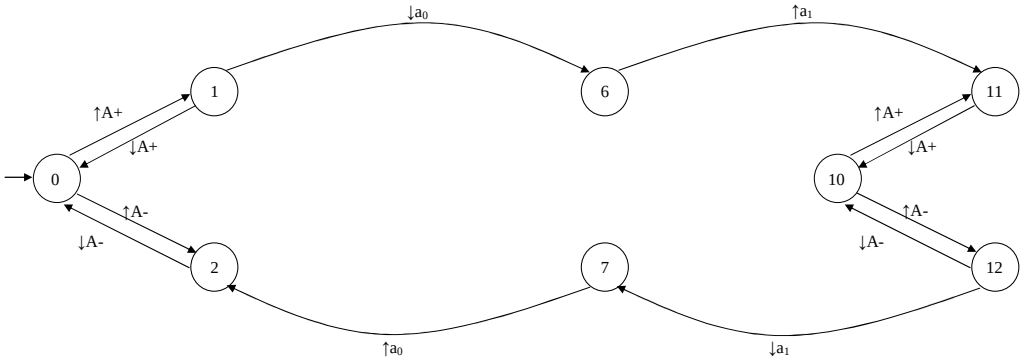


Figure 3-31 : Intégration de la contrainte de sécurité interdisant le déplacement du vérin A sans l'envoi d'un ordre à l'EPO_A

Les contraintes de vivacité représentent une spécification chronologique sur les événements et représentent une autorisation sur les événements de sortie d'un état. Ces contraintes consistent donc à vérifier la chronologie des événements que l'on souhaite appliquer au système. Dans le cas où un événement ne respecterait pas une de ces contraintes, il faudrait alors supprimer la transition correspondante et vérifier l'accessibilité des états du modèle.

Les contraintes de vivacité pour l'exemple du vérin A sont représentées dans le Tableau 3-5. Elles expriment des conditions sur le vecteur d'état permettant l'autorisation d'un événement. Par exemple, la contrainte n°1 exprime le fait que l'activation du détecteur a_0 n'est autorisée que pour un vecteur d'état (a_0, a_1, A+, A-) = (0, 0, 0, 1). Ainsi, à partir de l'état 0 du modèle d'EPO qui possède un vecteur d'état (a_0, a_1, A+, A-) = (1, 0, 0, 0), le vérin A est en position a_0 sans aucun ordre. Il doit donc vérifier la contrainte n°5 qui autorise, à partir de cet état, uniquement l'activation de l'ordre ↑A+. Par conséquent, les autres événements sortant de cet état sont supprimés car non désirés. C'est le cas de l'événement ↑A- qui permettait d'aller vers l'état 2. A partir de chaque état, il faut vérifier les contraintes de vivacité correspondantes. L'$EPOC_A$ désiré pour le vérin A est au final obtenu lorsque toutes les contraintes sont satisfaites (Figure 3-24a).

Contraintes de vivacité n°	Condition sur la contrainte	Autorisation de l'événement
1	$\neg a_0 \wedge \neg a_1 \wedge \neg A+ \wedge A- = 1$	$\uparrow a_0$
2	$a_0 \wedge \neg a_1 \wedge A+ \wedge \neg A- = 1$	$\downarrow a_0$
3	$\neg a_0 \wedge \neg a_1 \wedge A+ \wedge \neg A- = 1$	$\uparrow a_1$
4	$\neg a_0 \wedge a_1 \wedge \neg A+ \wedge A- = 1$	$\downarrow a_1$
5	$a_0 \wedge \neg a_1 \wedge \neg A+ \wedge \neg A- = 1$	↑A+
6	$\neg a_0 \wedge a_1 \wedge A+ \wedge \neg A- = 1$	↓A+
7	$\neg a_0 \wedge a_1 \wedge \neg A+ \wedge \neg A- = 1$	↑A-
8	$a_0 \wedge \neg a_1 \wedge \neg A+ \wedge A- = 1$	↓A-

Tableau 3-5 : Contraintes de vivacité du vérin A

La modélisation des spécifications par équations logiques semble plus facilement interprétable que la modélisation par automates. En effet, la difficulté majeure de la modélisation par automates réside dans la traduction de ce que l'on souhaite exprimer par des automates à états. Les équations logiques sont obtenues plus facilement sans le passage par des étapes demandant une utilisation d'outils de composition synchrone. Il reste néanmoins à vérifier, dans les deux cas, que l'ensemble des contraintes a bien été appliqué.

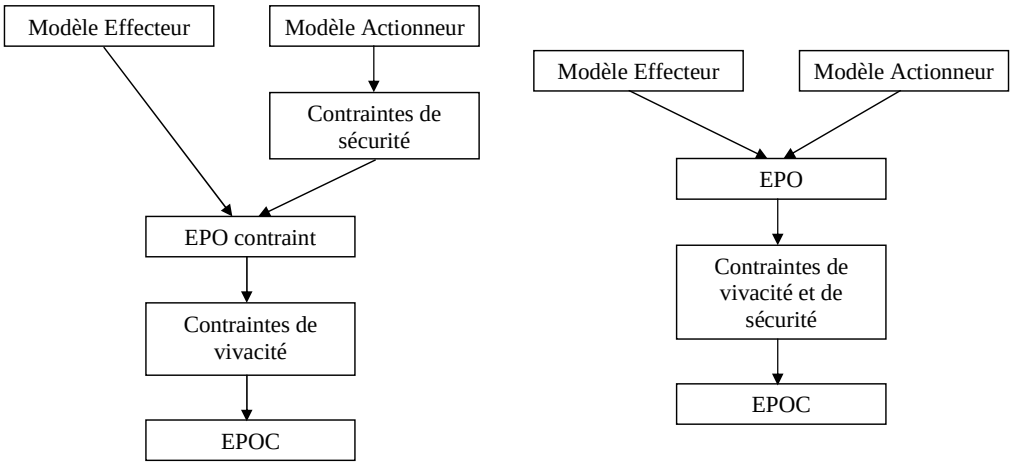
3.3.3. Discussion sur l'outil de spécification de la commande

L'intégration de la commande est une information primordiale pour le diagnostic puisqu'elle représente le comportement désiré de fonctionnement du procédé. Nous venons de voir que les spécifications de la commande peuvent être intégrées aux EPO soit par une modélisation en GRAFCET, soit sous forme de contraintes de sécurité et de vivacité, modélisées par des automates ou des équations logiques. Mais au final, que choisir pour l'intégration de la commande afin de réaliser le diagnostic des défaillances ?

Il apparaît clairement que l'intégration de l'information provenant de la commande à travers des contraintes de spécifications implique de n'oublier aucune de ces contraintes. Qu'elles soient modélisées par des automates ou par des équations logiques, l'élaboration des contraintes demande une certaine expérience sur le procédé. A contrario, il ne faut pas non plus ajouter trop de contraintes qui risquent de créer des redondances et/ou des incohérences entre elles ou pire encore des restrictions sur le système. L'utilisation des contraintes est donc très intéressante si l'on est capable d'exprimer toutes les spécificités du cahier des charges et notamment l'aspect séquentiel des événements.

En ce qui concerne le GRAFCET, il permet de garder tous les aspects du cahier des charges en terme de sûreté de fonctionnement en intégrant les contraintes de vivacité et de sécurité. Cependant, pour réaliser notre intersection, il est primordial d'établir l'hypothèse que la modélisation par GRAFCET soit robuste et fiable. Pour cela, nous utilisons les travaux réalisés sur la synthèse de commande de [NDJ 99], [PHI 03] et [TAJ 05] fournissant une commande dite optimale. Cette particularité nous fait dire que le GRAFCET est très intéressant dans le cadre de nos travaux à condition que les différents modèles soient validés par l'expert avant utilisation. Le GRAFCET nous paraît donc l'outil le plus approprié à l'intégration de la commande. Cependant, cette méthode d'intégration peut poser certains problèmes au niveau du partage de ressources communes. En effet, lorsqu'un détecteur est utilisé pour deux EPO, il faut tenir compte des événements communs aux langages des deux EPO ce qui n'est pas toujours évident à réaliser. Dans ce cas, le compromis est d'utiliser les contraintes en sus du GRAFCET afin de vérifier chaque modèle. Ainsi, le GRAFCET extrait le comportement désiré localement sur chaque EPO et les contraintes vérifient le respect de ce comportement désiré localement.

Une autre discussion concerne l'intégration de certaines contraintes, notamment celles de sécurité. En effet, elles peuvent être considérées dès le modèle actionneur ce qui a pour effet de diminuer le nombre d'états sur le modèle d'EPO (Figure 3-32a). Pour notre étude, nous avons souhaité décrire les EPO comme complètement indépendant de la commande (Figure 3-32b).

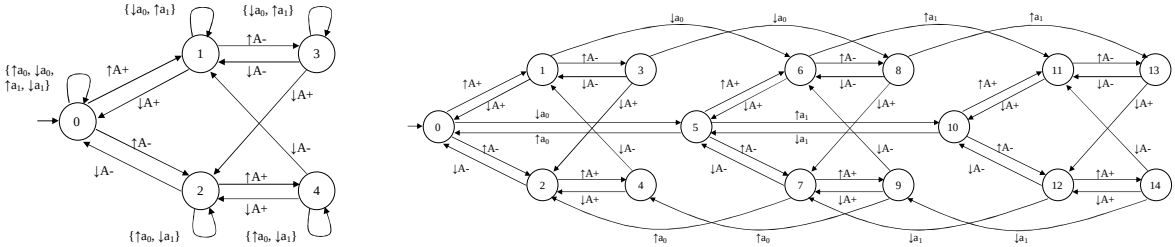


a) Contraintes de sécurité sur l'actionneur b) Contraintes de sécurité sur l'EPO

Figure 3-32 : Positionnement des contraintes de sécurité

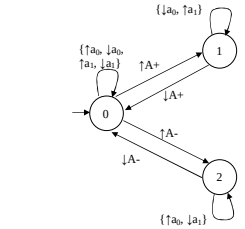
Afin de justifier ce choix, reprenons l'exemple de l'actionneur de la Figure 3-33a. Les états 3 et 4 représentent un comportement dangereux indiquant l'activation de deux ordres antagonistes. La description complète de l'EPO conserve alors cette possibilité d'activation d'ordres contraires par les états 3, 4, 8, 9, 13 et 14 de la Figure 3-33b obtenu par composition synchrone avec le modèle des détecteurs. Si la contrainte de sécurité "Ne pas activer les 2 ordres en mêmes temps" est intégrée dès le modèle actionneur (Figure 3-33c), alors le modèle d'EPO comprend un nombre d'états et de transitions réduit (Figure 3-33d).

Cette intégration des contraintes permet une meilleure lisibilité mais implique également une perte d'information sur l'EPO. Perte d'information que nous ne souhaitons pas dans le cadre du diagnostic des défaillances.

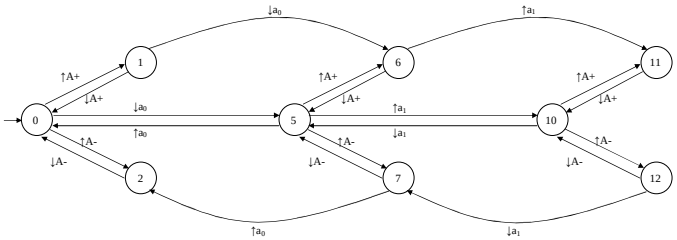


a) Modèle actionneur sans contrainte

b) Modèle PO sans contrainte



c) Modèle actionneur avec contrainte intégrée



d) Modèle PO avec contrainte intégrée

Figure 3-33 : Place de la contrainte de sécurité interdisant l'activation simultanée des ordres de sortie et de rentrée

Pour la suite, nous utilisons la structure de la Figure 3-32a pour l'intégration des contraintes. De plus, la constitution d'une bibliothèque d'EPO n'est pas compatible avec des contraintes propres à un cahier des charges particulier. En restant libre de toutes contraintes, nos éléments de la bibliothèque peuvent être ainsi utilisés sans a priori sur le cahier des charges.

3.4. Cohabitation des informations

Le diagnostiqueur détecte un défaut soit par observation d'une évolution non attendue, soit par observation de la violation d'une contrainte temporelle correspondant à la dynamique du procédé. Le premier cas regroupe les défauts correspondant à des incohérences logiques et séquentielles. Les incohérences logiques se traduisent par le fait qu'un actionneur ne peut se trouver dans deux positions différentes (deux capteurs de fin de course à 1) alors que les incohérences séquentielles consistent à respecter un ordre d'événements qui se succèdent (l'événement *a* se produit avant l'événement *b*). Le second cas correspond à un critère temporel qui observe les caractéristiques temporelles du procédé et détecte toute violation de contrainte temporelle de type : "l'événement *a* doit avoir lieu dans l'intervalle [durée minimale, durée maximale]".

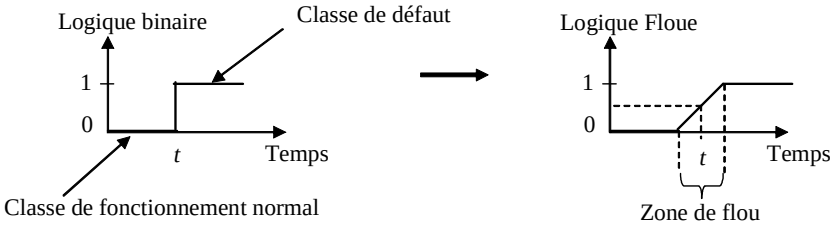
L'obtention des EPOC a permis d'intégrer, ou de faire cohabiter, l'information de la PO avec l'information représentée par la commande. Dans ce paragraphe, nous proposons d'intégrer aux EPOC, l'information temporelle à travers des fonctions de prévision pour constituer les diagnostiqueurs locaux.

3.4.1. Intégration de l'information temporelle

a) Principe

Les automates, représentant les EPOC, constituent la base des diagnostiqueurs locaux. Partant du principe que tout ce qui n'est pas normal est anormal, le diagnostic des défaillances que nous proposons consiste à s'assurer que le système ne dévie pas de son comportement désiré. Afin d'enrichir l'information présente sur le système, une information temporelle sur la réactivité des actionneur est ajoutée. Cette information consiste à définir le temps de réaction d'un actionneur mesuré par des capteurs, suite à l'envoi d'un ordre. En effet, l'envoi d'un ordre de la commande engendre une réaction de l'actionneur qui va changer de position. Ce changement est caractérisé par l'arrivée d'un nouvel événement comme, par exemple, le passage de "1" à "0" d'un capteur. Nous insistons sur le fait qu'ici nous ne parlons pas de temps de communication entre la PC et la PO, que nous considérons comme nul, mais bien du temps de réaction entre les événements représentant la dynamique du procédé.

L'évaluation quantitative du critère temporel donne un résultat en terme d'appartenance à une des deux classes : la classe du fonctionnement normal et la classe des défauts. Si cette fonction d'appartenance est considérée comme booléenne alors cette évaluation prend une des deux valeurs : "0" pour la classe normale, indiquant le respect de la contrainte temporelle, ou "1" pour la classe de défauts, indiquant une violation de cette contrainte (Figure 3-34a). Nous proposons d'évaluer ce critère temporel en utilisant la logique floue qui permet de graduer l'appartenance à une de ces deux classes par un nombre réel compris entre 0 et 1. L'augmentation de la valeur de la fonction d'appartenance de 0 vers 1 signifie une évolution du fonctionnement du procédé vers un défaut. Cette évolution caractérise un défaut de type progressif qui peut être l'indicateur d'une prévision de l'occurrence d'un défaut (Figure 3-34b). Pour chaque actionneur, il est possible d'établir des temps de réponse par apprentissage ou par des équations mathématiques selon la technologie utilisée. Ainsi, à partir du temps *t* calculé, l'opérateur humain peut y définir une zone floue définissant un intervalle de tolérance. Cet intervalle permet d'informer l'utilisateur sur une dérive du système et de réaliser ainsi un pronostic sur l'état d'usure de l'élément [PHI 05b]. Par la suite, nous considérerons que ces intervalles nous sont donnés.



a) Temps de réponse binaire

b) Temps de réponse évalué par une fonction floue

Figure 3-34 : Représentation du temps de réponse

Prenons le cas du vérin A de l'exemple de la Figure 3-13, son temps de sortie est évalué grâce aux paramètres donnés par le constructeur. Il est de 30 secondes pour une course de un mètre. En prenant un intervalle de [30s, 33s], on tient compte d'une marge de sécurité de 10%, et on dispose ainsi du temps de réaction au plus tard. Au même titre que le temps de réaction au plus tard, nous définissons également un temps de réaction au plus tôt [27s, 30s]. En effet, la course d'un vérin ou la rotation d'un moteur trop rapide peut engendrer de graves problèmes de casse mécanique.

Dès lors, au même titre que les chroniques ou les *templates* [BOU 03a], [HOL 94], nous définissons une fonction de prévision floue pour chaque état de chaque $EPOC_i$. Cette fonction de prévision *FP* est déclenchée suite à l'occurrence d'un événement α_1 en attente d'un événement d'arrivée α_2 (Figure 3-35).

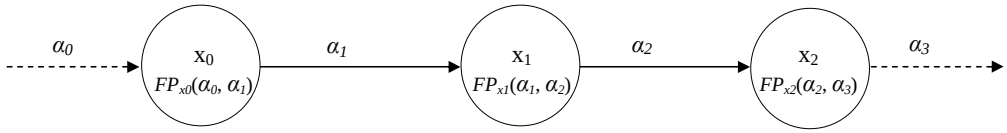


Figure 3-35 : Représentation de la fonction de prévision sur chaque état d'un $EPOC_i$

b) Formalisation des fonctions de prévision

Pour la Figure 3-35, le délai entre deux événements α_1 et α_2 est défini sur l'état 1 par la fonction de prévision $FP(\alpha_1, \alpha_2)$ où α_1 est l'événement entrant qui enclenche une temporisation d'attente de l'événement α_2. Pour cette fonction, deux intervalles sont établis, le premier correspond au délai minimal accepté pour l'occurrence du prochain événement attendu $[t_{min1}, t_{min2}]$. Le second intervalle correspond au délai maximal accepté $[t_{max1}, t_{max2}]$ (Figure 3-36).

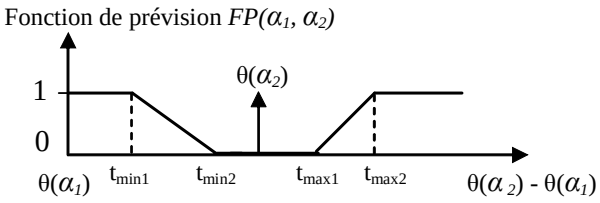


Figure 3-36 : Fonction de prévision

La fonction de prévision informe l'utilisateur sur le temps de réaction de l'actionneur. Elle est définie par $FP(\alpha_1, \alpha_2) = \{\alpha_1, x_1, (\alpha_2, [t_{min1}, t_{max2}], x_2, l_1)\}$ où l'état x_1 attend l'événement α_2 dans l'intervalle $[t_{min1}, t_{max2}]$ pour atteindre l'état x_2 avec une fonction de décision l_1. Cette fonction de décision précise si la contrainte temporelle n'a pas été satisfaite en donnant les étiquettes des partitions de défauts responsables de cette non-satisfaction.

La fonction de prévision peut prendre toutes les valeurs entre 0 et 1, elle est définie par :

$$\forall \alpha_1, \alpha_2 \in \Sigma_o, FP(\alpha_1, \alpha_2) = \begin{cases} 1 \text{ si } \tau < t_{\min 1} \\ \dfrac{\tau - t_{\min 2}}{t_{\min 1} - t_{\min 2}} \text{ si } t_{\min 1} \le \tau < t_{\min 2} \\ 0 \text{ si } t_{\min 2} \le \tau \le t_{\max 1} \\ \dfrac{\tau - t_{\max 1}}{t_{\max 2} - t_{\max 1}} \text{ si } t_{\max 1} < \tau \le t_{\max 2} \\ 1 \text{ si } \tau > t_{\max 2} \end{cases}$$

avec $\tau = \theta(\alpha_2) - \theta(\alpha_1)$ le délai entre la date d'occurrence $\theta(\alpha_1)$ de l'événement α_1 et la date d'occurrence $\theta(\alpha_2)$ de l'événement α_2. Ainsi, si aucun événement n'est survenu après t_{max2}, il n'est plus nécessaire d'attendre l'occurrence de α_2 pour dire que le système est en défaut.

On définit par $FP_x = \{FP(\alpha_i, \alpha_j), FP(\alpha_k, \alpha_h), \ldots\}$ l'ensemble des fonctions de prévision affecté sur l'état x. En effet, il est possible d'avoir plusieurs fonctions de prévision pour le

même état. Cette situation peut se retrouver dans les cas d'indéterminisme liés à l'occurrence de plusieurs événements sortant d'un même état et aboutissant à des états distincts. Ceci peut également se produire dans le cas où plusieurs événements, entrant dans un même état, enclencheraient une fonction de prévision appropriée.

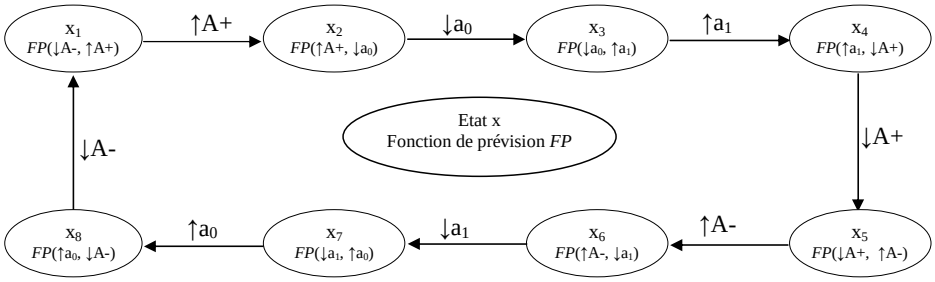
Par conséquent, la détection d'un défaut sur un état x se fait par la fonction *max*($FP(\alpha_i, \alpha_j)$, $FP(\alpha_k, \alpha_h)$, …) noté $maxFP_x$. Ainsi, si la fonction de prévision de l'état x_1 de la Figure 3-35 est satisfaite, $maxFP_{x1} < 1$, alors aucune étiquette n'est retournée puisque le fonctionnement est normal. Par contre, si elle n'est pas respectée, la fonction de prévision FP_{x1} déclare alors un défaut, $maxFP_{x1} = 1$, et retourne une étiquette de défaut. L'intégration des fonctions de prévision sur chaque état de chaque $EPOC_i$ conduit à l'obtention des EPOC Temporels (EPOCT).

Les fonctions de prévision évaluent des contraintes temporelles impliquant des événements observables localement par un élément de la partie opérative EPO_i.

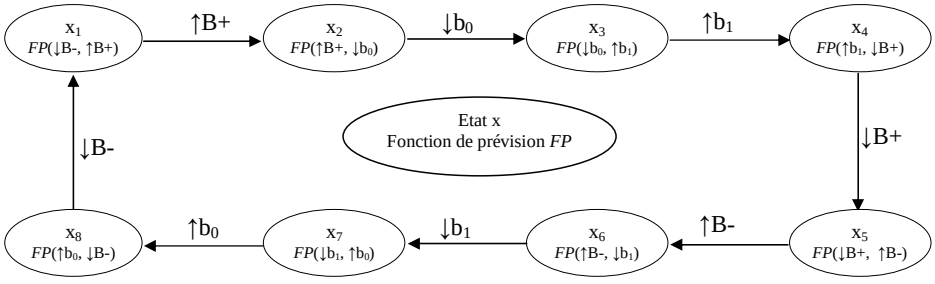
Malgré l'intérêt de l'introduction de la logique floue, il est très difficile de mesurer tous les temps de réponse pour chaque événement, et cela même par apprentissage. Les fonctions floues présentées ici requièrent une étude importante à réaliser en amont avec l'aide de l'expert, des différentes documentations techniques et d'un apprentissage rigoureux. Par la suite, nous considérerons que les différents temps qui constituent les fonctions *FP* nous sont donnés.

c) Application à l'exemple

Pour l'exemple du vérin A du système de transfert de pièces de la Figure 3-13, à partir de l'$EPOC_A$ obtenu en Figure 3-24a, il est possible de définir une fonction de prévision pour chaque état. Il en résulte le modèle de l'Elément de Partie Opérative Commandé Temporel du vérin A ($EPOCT_A$) de la Figure 3-37a. L'$EPOCT_B$ du vérin B est donnée en Figure 3-37b.



a) Elément de Partie Opérative Commandé Temporel du vérin A : $EPOCT_A$



b) Elément de Partie Opérative Commandé Temporel du vérin B : $EPOCT_B$

Figure 3-37 : Elément de Partie Opérative Commandé Temporel des vérins A et B

A partir de l'état x_2, il est possible de créer la fonction de prévision de l'état x_3 par $FP_{x3} = FP(\downarrow a_0, \uparrow a_1) = \{\downarrow a_0, x_3, (\uparrow a_1, [5s, 15s], x_4, F4)\}$. Cette fonction de prévision indique que, suite au passage à 0 du détecteur a_0, le détecteur a_1 doit passer à 1 dans l'intervalle de temps [5s, 15s] pour arriver à l'état x_4. Dans le cas où la fonction ne serait pas satisfaite, la fonction de décision retournerait l'étiquette F4 indiquant qu'un défaut issu de la partition Π_{F4} est survenu.

Pour cet exemple, nous avons considéré qu'il était possible d'établir une fonction de prévision pour chaque état, ce qui n'est pas toujours possible.

3.4.2. Intégration d'un super état de défaut

La cohabitation de l'ensemble des informations provenant de la PO, de la PC et de la réactivité temporelle des actionneurs, a été réalisée par la construction de l'EPOCT. A partir de ces modèles, il faut maintenant être capable de détecter l'ensemble des défauts susceptibles de survenir avant de pouvoir les identifier par les diagnostiqueurs. Pour cela, nous avons créé, pour chacun des EPOCT, un super état de défaut atteint suite à l'occurrence de n'importe quel défaut.

a) Principe

L'ajout d'un super état de défaut à partir des états de l'$EPOCT_i$ permet d'obtenir des EPOCT étendus. Ce super état de défaut est atteint soit par l'occurrence d'un événement non attendu, soit par la violation d'une fonction de prévision. Il permet de détecter un défaut sans en donner la cause. Un $EPOCT_i$ étendu est défini par $(X_i \cup \{X_{Fi}\}, \Sigma_{io}, \delta_i, x_{i0}, V_i, h_i, FP_{xj})$ où :

- $\{X_{Fi}\}$ est le super état de défaut X_{Fi},
- $X_i \cup \{X_{Fi}\}$ est l'ensemble des états de l'$EPOCT_i$ étendu,
- Σ_{io} est l'ensemble des événements observables par l'élément de partie opérative i,
- δ_i représente la fonction de transition d'état $\delta_i : X_i \times \Sigma_i^* \rightarrow X_i \cup \{X_{Fi}\}$,
- x_{i0} correspond à l'état initial,
- V_i représente le vecteur d'entrée/sortie avec $V_i(x)$ le vecteur de l'état x,
- $h_i : X_i \cup \{X_{Fi}\} \rightarrow \Sigma_{io}$ est la fonction événementielle de sortie où $h_i(x)$ est l'événement observable à la sortie de l'état x,
- FP_{xj} représente l'ensemble des fonctions de prévision de l'état x_j modélisant les contraintes temporelles entre les événements entrant et sortant de l'état x_j.

Dès lors, pour chaque état $x \in X_i$, un vecteur d'état est défini. Ce vecteur $V_i(x) = (e_1, e_2, \ldots, e_p, \ldots, e_d)$ contient l'ensemble des d variables d'état représentant les capteurs et les ordres.

L'occurrence d'un événement observable $h_i(x)$ à partir de x conduira à l'état x' avec un vecteur d'état $V_i(x') = (e'_1, e'_2, \ldots, e'_p, \ldots, e'_d)$. Ce dernier peut être calculé en utilisant un vecteur de déplacement booléen $Z_{hi(x)} = (z_1, z_2, \ldots, z_p, \ldots, z_d)$ dans IB^d. Ce vecteur décrit l'influence de $h_i(x)$ sur $V_i(x)$. Si $z_p = 1$, alors la valeur de la $p^{ième}$ variable de $V_i(x)$ est mise à 1 ou à 0 lorsque l'événement $h_i(x)$ arrivera. Par contre, si $z_p = 0$, alors la valeur de la $p^{ième}$ variable de $V_i(x)$ ne change pas lorsque l'événement $h_i(x)$ arrivera. Cela peut être défini formellement par :

$$\forall x, x' \in X_i \cup \{X_{Fi}\} : \delta_i(x, h_i(x)) = x' \Rightarrow V_i(x') = V_i(x) \oplus Z_{hi(x)} \qquad \text{(équation 1)}$$

De la même manière, nous pouvons définir les vecteurs de déplacement pour les autres événements. L'ensemble de tous les vecteurs de déplacement de tous les événements du procédé génère une matrice de déplacement *Z*.

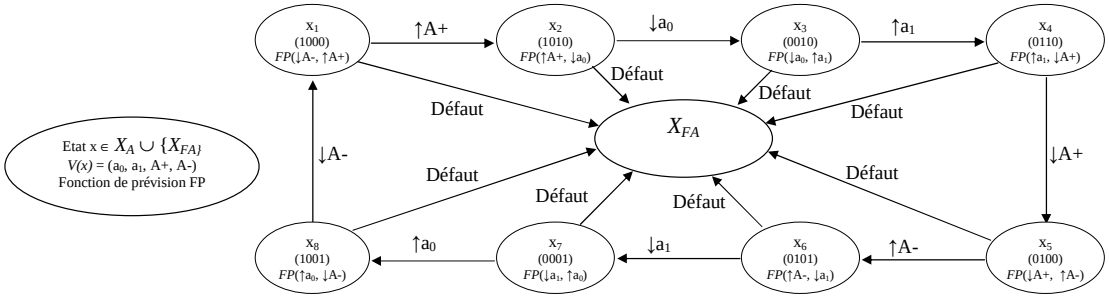
Cependant, tous les événements observables ne doivent pas survenir à partir d'un état. Ainsi, il faut définir à partir de chaque événement observable, $h_i(x) \in \Sigma_{io}$, une condition d'autorisation, $en_{hi(x)}(x) \in \{0, 1\}$. Cette condition d'autorisation permet d'indiquer si cet événement peut survenir à partir de l'état x ou non. Par conséquent, à partir d'un état x, il est possible d'avoir l'événement $h_i(x)$ que si $en_{hi(x)}(x) = 1$. Si la condition d'autorisation $en_{hi(x)}(x)$ est égale à 0, alors cela signifie que l'événement survenu est un événement non attendu qui doit, par conséquent, atteindre le super état de défaut. L'équation 1 devient alors :

$$\forall x, x' \in X_i \cup \{X_{Fi}\} : \delta_i(x, h_i(x)) = x' \Rightarrow V_i(x') = V_i(x) \oplus (Z_{hi(x)}.en_{hi(x)}(x)) \qquad \text{(équation 2)}$$

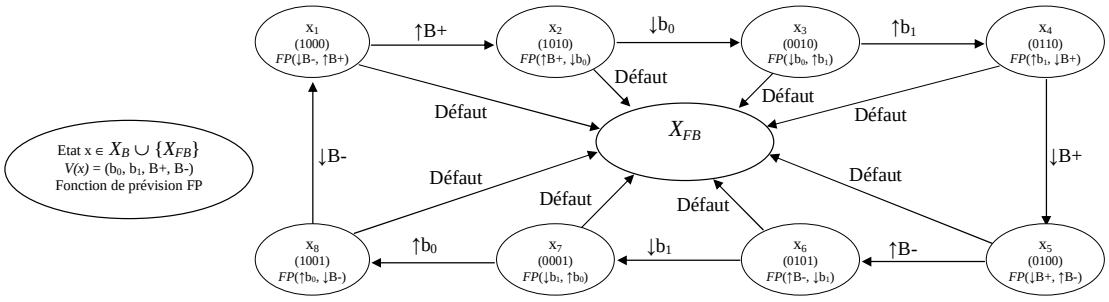
Le vecteur d'état $V_i(x)$ exprime une propriété statique des signaux logiques émis par les capteurs et/ou par la PC. Ce vecteur peut être utilisé pour détecter les incohérences logiques violant cette propriété statique. L'automate exprime la propriété dynamique du procédé représentée par le séquencement d'événements. Le non-respect de ce séquencement viole cette propriété dynamique indiquant l'occurrence d'un défaut dû à une incohérence séquentielle.

b) Application à l'exemple

En reprenant les éléments de partie opérative commandés temporels du vérin A et du vérin B ($EPOCT_A$ et $EPOCT_B$) de la Figure 3-37, nous obtenons, en ajoutant un super état de défaut X_{FA} et X_{FB}, les $EPOCT_A$ et $EPOCT_B$ étendus (Figure 3-38a et Figure 3-38b).



a) $EPOCT_A$ étendu du vérin A



b) $EPOCT_B$ étendu du vérin B

Figure 3-38 : $EPOCT_A$ et $EPOCT_B$ étendus des vérins A et B

Tout comportement ne respectant pas le comportement désiré des EPOCT par le séquencement des événements, par les sorties observables des détecteurs et des ordres, et par le temps de réponse évalué par les fonctions de prévision, doit entraîner une transition vers un super état indiquant l'occurrence d'un défaut.

Pour le vérin A, le vecteur d'état $V_A(x) = (a_0, b_0, A+, A-)$ s'exprime pour l'état x_1 par $V_A(x_1) = (1\ 0\ 0\ 0)$ avec la fonction de prévision $FP_{x1} = FP(\downarrow A-, \uparrow A+) = \{\downarrow A-, x_1, (\uparrow A+, [1s, 2s], x_2, l_{x1})\}$. L'événement sortant de cet état est $h_A(x_1) = \uparrow A+$. La matrice de déplacement, Z_A, du vérin A est donnée dans le Tableau 3-6 tandis que les conditions d'autorisation pour les événements du vérin A sont dans le Tableau 3-7.

Dès lors, d'après le Tableau 3-7 et le vecteur d'état $V_A(x_1) = (1\ 0\ 0\ 0)$, $\uparrow A+$ est le seul événement autorisé à partir de l'état x_1. D'après la matrice de déplacement Z_A, Tableau 3-6, son vecteur de déplacement est $Z_{\uparrow A+} = (0\ 0\ 1\ 0)$. Par conséquent, le vecteur d'état de x_2 est :

$V_A(x_2) = \nabla(V_A(x_1), h_A(x_1)) = \nabla((1\ 0\ 0\ 0), \uparrow A+) = (1\ 0\ 0\ 0) \oplus ((0\ 0\ 1\ 0).(1)) = (1\ 0\ 1\ 0)$.

Par conséquent, l'occurrence de tout autre événement que $\uparrow A+$, conduit au super état de défaut X_{FA}.

Variables d'états	$\uparrow a_0$	$\downarrow a_0$	$\uparrow a_1$	$\downarrow a_1$	$\uparrow A+$	$\downarrow A+$	$\uparrow A-$	$\downarrow A-$
a_0	1	1	0	0	0	0	0	0
a_1	0	0	1	1	0	0	0	0
A+	0	0	0	0	1	1	0	0
A-	0	0	0	0	0	0	1	1

Tableau 3-6 : Matrice de déplacement Z_A du vérin A

Evénement : $h_i(x)$	Condition d'autorisation : $en_{hi(x)}$
$\uparrow a_0$	$\neg a_0 \wedge \neg a_1 \wedge \neg A+ \wedge A- = 1$
$\downarrow a_0$	$a_0 \wedge \neg a_1 \wedge A+ \wedge \neg A- = 1$
$\uparrow a_1$	$\neg a_0 \wedge \neg a_1 \wedge A+ \wedge \neg A- = 1$
$\downarrow a_1$	$\neg a_0 \wedge a_1 \wedge \neg A+ \wedge A- = 1$
$\uparrow A+$	$a_0 \wedge \neg a_1 \wedge \neg A+ \wedge \neg A- = 1$
$\downarrow A+$	$\neg a_0 \wedge a_1 \wedge A+ \wedge \neg A- = 1$
$\uparrow A-$	$\neg a_0 \wedge a_1 \wedge \neg A+ \wedge \neg A- = 1$
$\downarrow A-$	$a_0 \wedge \neg a_1 \wedge \neg A+ \wedge A- = 1$

Tableau 3-7 : Conditions d'autorisation $en_{hi(x)}$ pour chaque événement $h_i(x)$ du vérin A

3.5. Construction des diagnostiqueurs locaux

Chaque diagnostiqueur D_i peut être obtenu à partir de chaque $EPOCT_i$ étendu en ajoutant une fonction de décision, $l_x \in \Delta$, à chaque état x de l'$EPOCT_i$ étendu. Cette fonction donne une indication sur la situation normale ou défaillante du procédé à travers une ou plusieurs étiquettes.

Un diagnostiqueur local est défini par $D_i = (X_i \cup \{X_{DFi}\}, \Sigma_{io}, \delta_i, x_{i0}, V_i, h_i, FP_j, l_i)$ où X_{DFi} est l'ensemble des états de défaut obtenu à partir de l'état de défaut X_{Fi} de l'$EPOCT_i$ étendu.

Par conséquent, l'état de défaut X_{Fi} de l'EPOCT$_i$ étendu devient un ensemble d'états de défaut X_{DFi} où chaque état est affecté à une fonction de décision l_{xi} indiquant la partition de défauts responsable de l'arrivée dans cet état défaillant.

3.5.1. Association des étiquettes aux états des diagnostiqueurs

Prenons l'exemple d'un diagnostiqueur D = ($X \cup \{X_{DF}\}$, Σ_o, δ, x_0, V, h, FP_j, l) où chaque état x est composé d'un vecteur d'entrée/sortie $V(x)$ composé de 3 variables (e_1 e_2 e_3) et où l est la fonction de décision retournant, soit une étiquette de fonctionnement normal N, soit une étiquette correspondant à un fonctionnement défaillant {F1, F2, F3} (Figure 3-39).

A partir de l'état normal x_0, avec le vecteur d'état $V(x_0)$ = (0 0 0) et la fonction de prévision $FP_{x0} = FP(\downarrow e_2, \uparrow e_1)$, l'événement observable $h(x_0) = \uparrow e_1$ est la seule évolution normale du procédé et conduit le procédé à l'état normal x_1 associé à une étiquette N. Par conséquent, les défauts susceptibles d'être détectés et identifiés à partir de l'état x_0 sont définis par :

- $h(x_0) = \uparrow e_2$ représentant un défaut observable sur e_2 associé à une étiquette F1 (état x_2),
- $h(x_0) = \uparrow e_3$ représentant un défaut observable sur e_3 associé à une étiquette F2 (état x_3)
- $FP_{x0} = 1$ représentant un défaut non observable sur e_1 associé à une étiquette F3. Ce défaut est détecté par la fonction de prévision FP_{x0} qui n'est pas satisfaite et qui atteint l'état de défaut x_4 par un événement de violation de fonction de prévision e_{FPx0}.

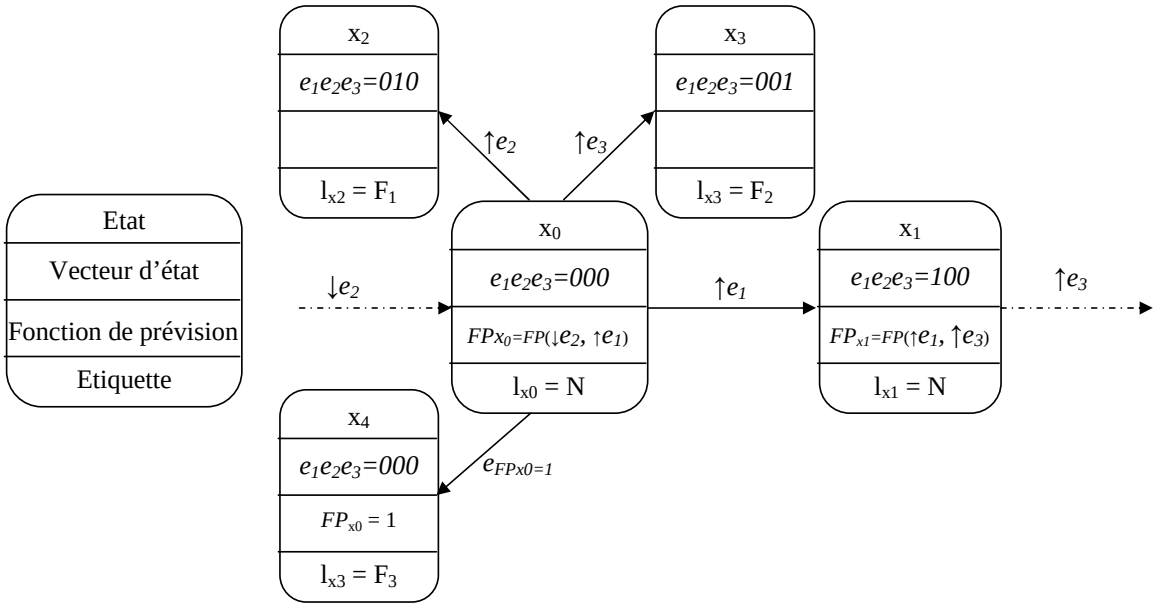


Figure 3-39 : Composants d'un état du diagnostiqueur D

Cependant, pour réaliser l'association des étiquettes de défauts aux états des diagnostiqueurs locaux, il faut au préalable définir l'ensemble des partitions de défauts que l'on souhaite diagnostiquer.

3.5.2. Détermination des partitions de défauts et de leurs étiquettes associées

a) Principe

Nous distinguons trois types de défauts :

- Les défauts des capteurs discrets appelés détecteurs,
- les défauts des actionneurs,
- les erreurs des ordres envoyés en ligne par la partie commande à la partie opérative.

Nous considérons que si un défaut est observable, sa détection est une tâche triviale. S'il est non observable, il faut inférer son occurrence en utilisant les événements observables et/ou les sorties des capteurs et les ordres de la PC. Le défaut observable se caractérise par le passage inattendu de 0 à 1 ou de 1 à 0 de la sortie d'un détecteur ou d'un ordre. Par contre, un défaut non observable se caractérise par un blocage de la sortie d'un détecteur au niveau 1 ou 0 ou par un actionneur qui reste bloqué malgré les ordres envoyés par la PC (Tableau 3-8).

Défaut détecteur observable	Passage inattendu d'un détecteur de 0 à 1
	Passage inattendu d'un détecteur de 1 à 0
Défaut détecteur non observable	Sortie d'un détecteur bloqué au niveau 0
	Sortie d'un détecteur bloqué au niveau 1
Erreur sur un ordre envoyé par la PC	Passage inattendu d'un ordre de 0 à 1
	Passage inattendu d'un ordre de 1 à 0
Erreur sur un ordre non envoyé par la PC	Attente d'activation d'un ordre non envoyée par la PC
	Attente de désactivation d'un ordre non envoyée par la PC
Défaut actionneur non observable	Actionneur bloqué inactif
	Actionneur bloqué actif

Tableau 3-8 : Types de défauts possibles

b) Application à l'exemple

Afin d'expliquer la construction des partitions de défauts pour chaque diagnostiqueur local, prenons l'exemple des différents défauts, Σ_{fA}, retrouvés sur le vérin A et récapitulés dans le Tableau 3-9.

Ils sont divisés en cinq partitions $\Sigma_{\Pi A} = \{\Pi_{F1}, \Pi_{F2}, \Pi_{F3}, \Pi_{F4}, \Pi_{F5}\}$:

- Les défauts sur le détecteur a_0 appartiennent à la partition $\Pi_{F1} = \{f_1, f_2, f_3, f_4\}$ et leur détection est indiquée par l'étiquette F1,
- les défauts liés à la sortie du vérin (erreur de l'ordre pour faire sortir le vérin) apparaissent dans la partition $\Pi_{F2} = \{f_5, f_6, f_7, f_8\}$ avec l'étiquette F2,
- les défauts sur l'actionneur lorsque le vérin est bloqué en position de rentrée ou en position de sortie se trouvent dans la partition $\Pi_{F3} = \{f_9, f_{10}\}$,
- les défauts sur le détecteur a_1 font partie de la partition $\Pi_{F4} = \{f_{11}, f_{12}, f_{13}, f_{14}\}$ et leur détection est indiquée par l'étiquette F4,

- les défauts liés à la rentrée du vérin (erreur de l'ordre pour faire rentrer le vérin) appartiennent à la partition $\Pi_{F5} = \{f_{15}, f_{16}, f_{17}, f_{18}\}$ avec l'étiquette F5.

F1	f_1	Passage inattendu de a_0 de 0 à 1	F4	f_{11}	Passage inattendu de a_1 de 0 à 1
	f_2	Détecteur a_0 bloqué à 0		f_{12}	Détecteur a_1 bloqué à 0
	f_3	Passage inattendu de a_0 de 1 à 0		f_{13}	Passage inattendu de a_1 de 1 à 0
	f_4	Détecteur a_0 bloqué à 1		f_{14}	Détecteur a_1 bloqué à 1
F2	f_5	Passage inattendu de A+ de 0 à 1	F5	f_{15}	Passage inattendu de A- de 0 à 1
	f_6	Passage inattendu de A+ de 1 à 0		f_{16}	Passage inattendu de A- de 1 à 0
	f_7	Activation de A+ non envoyée		f_{17}	Activation de A- non envoyée
	f_8	Désactivation de A+ non envoyée		f_{18}	Désactivation de A- non envoyée
F3	f_9	Actionneur bloqué inactif, en rentrée			
	f_{10}	Actionneur bloqué actif, en sortie			

Tableau 3-9 : Défauts pouvant avoir lieu sur le vérin A

Puisque les défauts observables peuvent être diagnostiqués trivialement par l'observation d'un événement inattendu et que les défauts non observables nécessitent l'utilisation de fonctions de prévision, il est donc possible de les classer par leur observation comme suit :

Défauts observables	Défaut détecteur observable	f_1, f_3, f_{11}, f_{13}
	Erreur sur un ordre envoyé par la PC	f_5, f_6, f_{15}, f_{16}
Défauts non observables	Défaut détecteur non observable	f_2, f_4, f_{12}, f_{14}
	Erreur sur un ordre non envoyé par la PC	f_7, f_8, f_{17}, f_{18}
	Défaut sur actionneur non observable	f_9, f_{10}

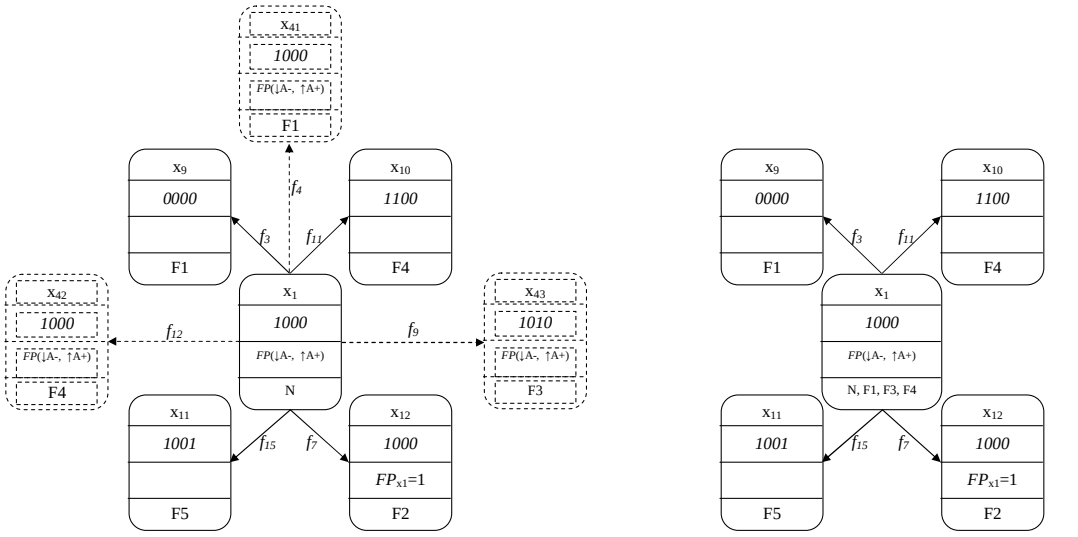
Tableau 3-10 : Classification des défauts par leur observation

La Figure 3-40a représente les défauts susceptibles de se produire à partir de l'état x_1 du diagnostiqueur D_A du vérin A de l'exemple de transfert de pièces de la Figure 3-13. A partir de l'état x_1 avec le vecteur d'état $V_A(x_1) = (a_0, a_1, A+, A-) = (1\ 0\ 0\ 0)$ et l'événement attendu $h_A(x_1) = \uparrow A+$, il est possible d'avoir :

- un défaut f_4 sur a_0 qui reste bloqué à 1, ce qui conduit à l'état x_{41} associé à l'étiquette F1,
- un défaut f_3 sur a_0 qui passe à 0, ce qui conduit à l'état x_9 associé à l'étiquette F1,
- un défaut f_{12} sur a_1 qui reste bloqué à 0, ce qui conduit à l'état x_{42} associé à l'étiquette F4,
- un défaut f_{11} sur a_1 qui passe à 1, ce qui conduit à l'état x_{10} associé à l'étiquette F4,
- un défaut f_7 correspondant à une erreur de la PC sur l'ordre A+ qui n'a pas été envoyé, ce qui conduit à l'état x_{12} associé à l'étiquette F2,
- un défaut f_{15} correspondant à une erreur de la PC sur l'ordre A- qui passe à 1, ce qui conduit à l'état x_{11} associé à l'étiquette F5,

- un défaut f_9 sur le vérin, qui reste bloqué à l'état inactif malgré un ordre A+, ce qui conduit à l'état x_{43} associé à l'étiquette F3.

Si, à partir de l'état x_1, le diagnostiqueur est capable de diagnostiquer, avec certitude, tous les défauts possibles, alors un état défaillant est associé à chaque partition de défauts. Par conséquent, l'état x_1 sera un état certain de fonctionnement normal associé à l'étiquette N (Figure 3-40a). Cependant, certains défauts ne sont pas diagnosticables à partir de l'état x_1. En effet, pour les défauts f_4 et f_{12}, le vecteur d'état et les fonctions de prévision, associées à x_1, ne permettent pas de réaliser une observation directe de ces défauts. Pour le défaut f_9, l'ordre a bien été envoyé et l'indique par un vecteur d'état qui évolue, $V_A(x_{43})$ = (1 0 1 0), cependant le vérin est bloqué et rien ne permet de le diagnostiquer à partir de x_1. Il faudra attendre une nouvelle observation ou la violation d'une fonction de prévision associée à un autre état pour diagnostiquer ces défauts. La fonction de décision de l'état x_1 comprend par conséquent, l'étiquette N et les étiquettes de défauts F1, F3 et F4 (Figure 3-40b).



a) Défauts susceptibles de se produire à partir de l'état x_1 du diagnostiqueur, D_A, du vérin A

b) Défauts détectables à partir de l'état x_1 du diagnostiqueur, D_A, du vérin A

Figure 3-40 : Défauts à partir de l'état x_1 du diagnostiqueur D_A du vérin A

3.5.3. Vers la construction des diagnostiqueurs locaux

3.5.3.1. Représentation des états des défauts détectables

Ce sont les fonctions de prévision qui permettent la détection des défauts non observables. En effet, une hypothèse forte sur les fonctions de prévision consiste à considérer l'intervalle de tolérance inférieur à la durée entre deux événements corrélés. Ainsi, il est possible de distinguer les défauts liés à un blocage par rapport aux défauts observables.

La Figure 3-41 représente un extrait montrant les évolutions possibles des défauts non observables. Par exemple, à partir de l'état x_1, le défaut f_{12}, représentant le détecteur a_1 bloqué, peut survenir et ne pas être détecté dans cet état. L'observation attendue de l'état x_3 après les événements ↑A+ et ↓a_0 est identique à celle qui a eu lieu réellement après un défaut. Dans ce cas, le système se trouve dans l'état x_{42} possédant le même vecteur d'état que l'état x_1. L'occurrence de l'événement ↑A+ conduit le système dans l'état de défaut x_{45} du diagnostiqueur D_A sans être détecté. Il faut attendre l'occurrence de l'événement ↓a_0 et le non

respect de la fonction de prévision de cet état pour détecter le défaut dans l'état x_{48}. Les états x_{42} et x_{45} ne permettent donc pas de détecter le défaut et ne sont donc pas essentiel au diagnostic puisque non observables. C'est le non-respect d'une fonction de prévision qui va permettre de détecter au plus tôt un défaut non observable à partir de l'état x_1. Il en est de même pour tous les défauts non détectables à partir de l'état x_1.

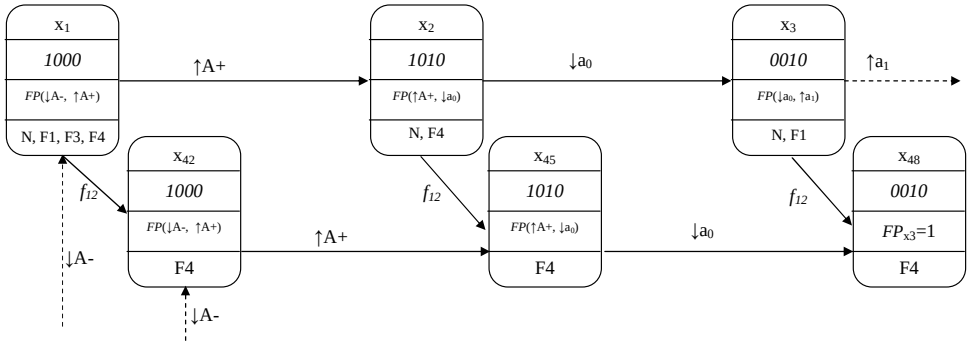


Figure 3-41 : Extrait du diagnostiqueur D_A du vérin A pour un défaut non observable f_{12}

Afin de ne représenter que l'information essentielle dans les diagnostiqueurs locaux, nous avons choisi de ne représenter les diagnostiqueurs locaux qu'à partir du comportement normal de l'élément et des états anormaux permettant la détection des défauts observables et non observables.

Pour l'exemple du diagnostiqueur local D_A du vérin A, un extrait est présenté Figure 3-42. Ce diagnostiqueur ne représente pas la date d'occurrence du défaut mais bien l'état à partir duquel il est possible de le diagnostiquer.

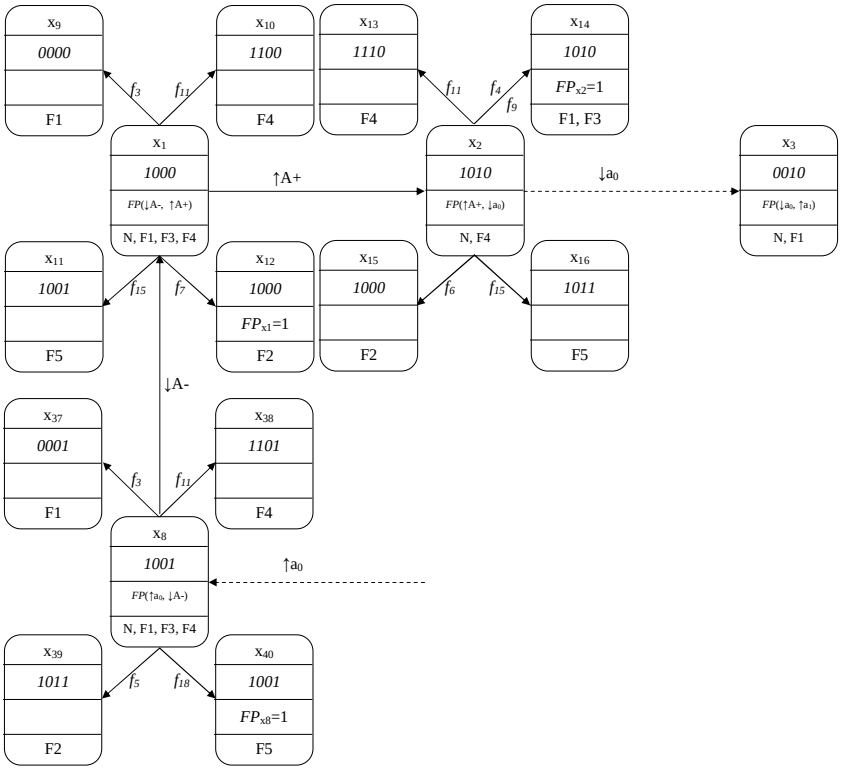


Figure 3-42 : Extrait du diagnostiqueur D_A du vérin A simplifié aux états permettant la détection des défauts

3.5.3.2. Diagnostic des défauts non observables

Les défauts observables représentent le changement inattendu d'une valeur d'un détecteur ou l'activation ou la désactivation inattendue d'un ordre de la PC. Leur détection est triviale et se réalise simplement avec l'opérateur logique "OU exclusif". Ainsi, si la valeur de l'opérateur logique "OU exclusif" est différente de 0, alors il est possible d'isoler la variable d'état qui ne correspond pas à ce qui est attendu dans les diagnostiqueurs locaux et de conclure que le dernier événement observé a déclenché une situation de défaut. L'évolution de la situation réelle doit être similaire à celle du comportement désiré du diagnostiqueur à chaque arrivée de nouvel événement. Dans le cas contraire, les défauts observables sont détectés dès leur apparition.

Pour les défauts non observables, à partir de chaque état d'un diagnostiqueur local, il n'est pas possible de détecter l'ensemble des défauts non observables mais uniquement ceux liés aux fonctions de prévision de cet état. Les fonctions de prévision étant affectées à des étiquettes de défauts, il est donc possible de localiser un état de défaut sur les diagnostiqueurs locaux lorsque ces fonctions sont violées.

Reprenons l'exemple du diagnostiqueur D_A du vérin A. A partir de l'état x_8, le défaut non observable f_{18} est détecté par la fonction de prévision $FP_{x8} = FP(\uparrow a_0, \downarrow A\text{-}) = \{\uparrow a_0, x_8, (\downarrow A\text{-}, [5s, 10s], x_1, F5)\}$. Le non-respect de cette fonction de prévision, $FP_{x8} = 1$, conduit le système dans l'état x_{40} qui possède le même vecteur d'état que x_8 mais l'étiquette indique bien un défaut appartenant à la partition F5.

Il est à noter que les défauts f_4, f_9, f_{12} non observables ne peuvent pas être détectés à partir de l'état x_8, mais rien ne permet de dire qu'ils ne sont pas survenus. Dès lors, l'état x_8 est associé aux étiquettes N, F1, F3 et F4 pour les défauts f_4, f_9, f_{12} non détectables à partir de cet état. Il est possible de représenter le diagnostiqueur D_A du vérin A de manière simplifiée en tenant compte des défauts non observables mais détectables par la Figure 3-43 afin de calculer le retard de détection nécessaire.

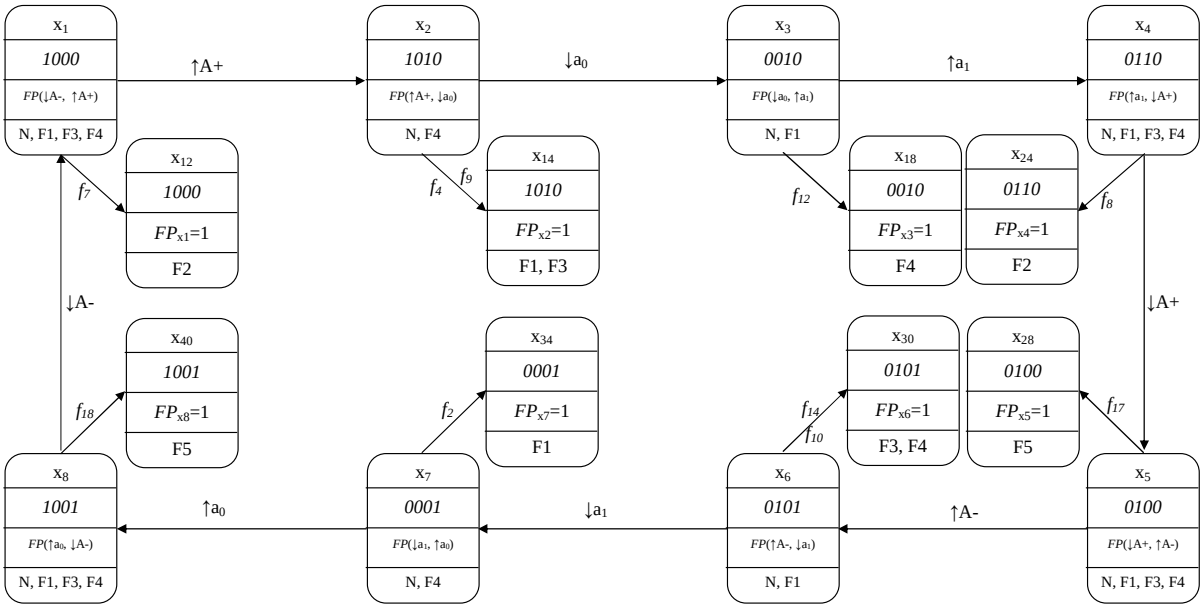


Figure 3-43 : Diagnostiqueur D_A simplifié aux défauts non observables

3.6. Vérification de la co-diagnosticabilité mixte d'une structure décentralisée

Nous venons de voir comment détecter et localiser les différents défauts par les diagnostiqueurs locaux. Maintenant, il faut s'assurer que l'ensemble de ces diagnostiqueurs locaux est capable de diagnostiquer l'ensemble des défauts comme un diagnostiqueur global. Il faut alors définir une notion de co-diagnosticabilité. Nous avons vu dans le chapitre 2 que la notion de diagnosticabilité a été développée dans le cadre d'une modélisation à base d'événements ou à base d'états. Cependant, les diagnostiqueurs utilisent une modélisation mixte (à base d'événements, à base d'états et à base d'informations temporelles). Ce paragraphe consiste par conséquent à proposer une notion de co-diagnosticabilité mixte pour notre approche de diagnostic décentralisée.

Tout d'abord, il est à noter que la notion de co-diagnosticabilité ne peut s'appliquer que pour la vérification de la propriété de diagnosticabilité des défauts non observables puisque les défauts observables sont détectables implicitement par leur observation. Cependant, nous avons vu que la détection des défauts non observables s'effectuait essentiellement par des fonctions de prévision associées à chaque état du diagnostiqueur local. Dès lors, s'il l'on est capable d'établir pour chaque état, toutes les fonctions de prévision possibles avec leurs intervalles de tolérance, alors la notion de co-diagnosticabilité est définie uniquement de manière temporelle ($maxFP_x$ = 1 et l_x = Fi). Cependant, il est très difficile de définir l'ensemble des fonctions de prévision pour un diagnostiqueur local. Par conséquent, il faut faire appel à l'observation d'événements corrélés ou aux vecteurs d'état de sortie des états des diagnostiqueurs locaux. La notion de co-diagnosticabilité mixte est ainsi développée.

3.6.1. Notion de co-diagnosticabilité mixte

Pour définir la notion de co-diagnosticabilité mixte, il faut tout d'abord constituer l'ensemble des états atteint par l'occurrence d'un défaut de type Fi. Pour cela, prenons un diagnostiqueur local $D_i = (X_i \cup \{X_{DFi}\}, \Sigma_{io}, \delta_i, x_{i0}, V_i, h, FP_j, l_i)$ avec X_{DFi} l'ensemble des états de défaut du diagnostiqueur D_i. Il est possible de définir par S_{Fi} l'ensemble des états atteint par l'occurrence d'un défaut de type Fi. Prenons H_{Fi} comme l'ensemble de tous les vecteurs d'états de l'ensemble S_{Fi}, alors la partition de sortie H_{Fi} peut être définie par :

$$\forall x \in S_{Fi},\ V' = V(x') \Rightarrow V' \in H_{Fi}$$

La problématique de la co-diagnosticabilité des défaillances est de diagnostiquer de façon certaine l'occurrence d'un défaut de type Fi appartenant à un comportement défaillant (L-K) $\cap\ \Psi(\Pi_{Fi})$ par au moins un diagnostiqueur local dans un délai fini. $\Psi(\Pi_{Fi})$ est l'ensemble de toutes les séquences d'événements qui se termine par un événement de défaut appartenant à Π_{Fi}. Par conséquent, la notion de co-diagnosticabilité mixte est définie par :

Soit un modèle du procédé *G* avec son langage L, *m* diagnostiqueurs locaux D_j ($j \in \{1, 2, \ldots, m\}$) et un langage de spécification K. Ce modèle est dit co-diagnosticable par rapport aux fonctions de projections P_{Lj} avec $j \in \{1, 2, \ldots, m\}$ et pour un ensemble de partition de défauts Π_{Fi} ($i \in \{1, 2, \ldots, r\}$) ssi :

$$\forall f \in \Pi_{Fi},\ \forall i \in \{1, 2, \ldots, r\},\ \exists n \in \aleph,\ \forall st \in (L\text{-}K) \cap \Psi(\Pi_{Fi}),\ \forall st \in L,\ |t| \geq n$$

$$\forall j \in \{1, 2, \ldots, m\}, \forall w \in P_{Lj}^{-1}(P_{Lj}(st)) \cap (L\text{-}K) \Rightarrow w \in (L\text{-}K) \cap \Psi(\Pi_{Fi})$$

$$\text{et/ou } \forall x \in X_j,\ x' = \delta(st, x),\ x'' = \delta(w, x),\ V' = V(x'),\ V'' = V(x'') \Rightarrow$$

$$V', V'' \in H_{Fi}$$

$$\text{et/ou } maxFP_x = 1 \text{ et } l_x = Fi$$

La vérification de cette notion signifie que tout défaut appartenant à la partition Π_{Fi} est co-diagnosticable par au moins un diagnostiqueur local D_j. En d'autres termes, la co-diagnosticabilité mixte signifie que le modèle d'un procédé *G* est co-diagnosticable par au moins un diagnostiqueur local D_j en utilisant au plus une de ces conditions :

- S'il existe une séquence d'événements *s* se terminant par un défaut de type Fi et n'appartenant pas au langage désiré, et une séquence d'événements *t* suffisamment longue continue à *s*, alors toute séquence d'événements *w* ayant le même comportement observable que celui de *st* par rapport à un diagnostiqueur D_j : $P_{Lj}(st) = P_{Lj}(w)$, doit contenir également un défaut de type Fi (co-diagnosticabilité à base d'événements).
- Si un état x' est atteint par une séquence d'événements contenant un défaut de type Fi et possède un vecteur d'état $V' = V(x')$ appartenant à la partition de sortie H_{Fi}, alors tout autre état atteint par une autre séquence d'événements contenant le même défaut de type Fi, doit avoir un vecteur d'état $V'' = V(x'')$ appartenant à la même partition de sortie H_{Fi} (co-diagnosticabilité à base d'états).
- Si une fonction de prévision définissant une contrainte temporelle entre l'occurrence des événements observables $P_L(st)$, n'est pas satisfaite en raison de l'occurrence d'un défaut de type Fi, alors toute séquence d'événements *w*, ayant le même comportement observable que celui de *st* par rapport au diagnostiqueur local D_j, doit avoir une fonction de prévision non satisfaite et fournir une fonction de décision correspond à l'étiquette de défaut, l_x = Fi.

3.6.2. Retard de détection

a) Principe

Afin de déterminer la co-diagnosticabilité d'un procédé, il faut pouvoir vérifier qu'un état certain est atteignable et cela dans un délai fini.

Un état certain est un état qui possède une fonction de décision empêchant toute ambiguïté de décision. De plus, le délai de détection d'un défaut doit être borné. Autrement dit, le diagnostiqueur ne doit pas posséder de boucles d'incertitude. Nous avons vu que les défauts observables sont directement diagnosticables sans retard de détection. Par contre, pour les défauts non observables, il faut pouvoir établir, à partir de chaque diagnostiqueur local, le délai de détection d'un défaut. En effet, le diagnostiqueur local permet de connaître le délai entre l'occurrence d'un défaut et sa détection [MOU 05]. Il est alors possible d'établir le retard de détection à un défaut comme suit :

Pour chaque diagnostiqueur local D_i, une matrice de défauts $MF_i(MxN)$ est construite, où M est le nombre d'événements de défauts non observables et N le nombre d'états normaux. Pour chaque état normal, si un défaut ne peut pas être détecté, il faut alors mettre un "0" dans la case $MF_{jj'}$ correspondant à cet état (colonne j') et à ce défaut (ligne j). Dans le cas contraire la valeur est "1". Le retard de détection, RD_{imax}, pour le diagnostiqueur local D_i correspond alors au nombre d'événements maximum qu'il faut avoir afin de détecter un défaut à partir de son occurrence en utilisant l'équation suivante :

$$RD_{imax} = max(NO0_j(MF_{jj'} : j' \in (1, 2,..., N)) : j \in (1, 2,\ldots, M))$$

où $NO0_j$ est le nombre de 0 successifs sur chaque ligne j, $MF_{jj'} \in \{0, 1\}$ est la valeur assignée à la ligne j et à la colonne j'. Cette matrice est définie pour chaque boucle dans l'$EPOCT_i$.

b) Application à l'exemple

A partir de l'état x_1 du diagnostiqueur D_A du vérin A (Figure 3-42), il est possible de diagnostiquer sans retard les défauts observables $\{f_1, f_3, f_5, f_6, f_{11}, f_{13}, f_{15}, f_{16}\}$ après l'observation d'un événement (Tableau 3-10). Par contre, pour les défauts non observables $\{f_2, f_4, f_7, f_8, f_9, f_{10}, f_{12}, f_{14}, f_{17}, f_{18}\}$, détectables par les fonctions de prévision, il faut construire la matrice de défauts permettant d'établir le retard de détection (Tableau 3-11). Le défaut f_2 ne peut pas survenir à partir des états x_1, x_2, x_7 et x_8. Les cases correspondantes à ces états, colonnes 1, 2, 7 et 8, et ligne 1, auront la valeur logique "1".

	Etat / Défaut	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
F1	f_2	1	1	0	0	0	0	1	1
	f_4	0	1	1	1	1	1	1	0
F2	f_7	1	1	1	1	1	1	1	1
	f_8	1	1	1	1	1	1	1	1
F3	f_9	0	1	1	1	1	1	1	0
	f_{10}	1	1	1	0	0	1	1	1
F4	f_{12}	0	0	1	1	1	1	0	0
	f_{14}	1	1	1	0	0	1	1	1
F5	f_{17}	1	1	1	1	1	1	1	1
	f_{18}	1	1	1	1	1	1	1	1

Tableau 3-11 : Matrice de défauts, MF_A, du vérin A

A partir de la matrice complète, il est possible alors de connaître, pour le vérin A, le retard de détection, RD_{Amax}. Par exemple, pour détecter un défaut de type F3, il faut attendre au plus deux événements observables. En effet, puisque le défaut f_{10} possède deux "0" qui se suivent à partir de l'état x_4, cela signifie que ce défaut ne peut pas être détecté avant l'état x_6. Ce défaut peut donc être diagnostiqué à la fin de l'occurrence de deux événements observables.

Pour détecter l'ensemble des défauts non observables sur le vérin A, le retard de détection RD_{Amax} est égal à 4 événements observables à cause des défauts f_2 et f_{12} qui possèdent quatre "0" successifs sur leur ligne de la matrice MF_A. Il en est de même pour le vérin B. Ainsi, il est possible de connaître le retard de détection maximum sur l'ensemble du procédé par $RD_{max} = max(RD_{Amax}, RD_{Bmax}) = max(4, 4) = 4$.

Il est à noter que les défauts non observables des partitions Π_{F2} et Π_{F5} sont directement détectables car ce sont des défauts provenant de la commande sur des ordres attendus. De plus, en regardant plus précisément la matrice de défaut MF_A du vérin A, on s'aperçoit que les colonnes permettent d'indiquer, pour chaque état, les défauts qu'il n'est pas possible de diagnostiquer. Par exemple, la matrice indique par les "0", présents sur la colonne de l'état x_4, que cet état ne permet pas de détecter les défauts de type F1, F3 et F4, ce qui justifie les étiquettes retrouvées dans la fonction de décision de l'état x_4 sur la Figure 3-43.

Les fonctions de prévision permettent de diagnostiquer avec un retard minimal les défauts non observables. Sans elles, ces défauts seraient détectés par l'observation d'un événement corrélé mais avec un retard plus important. En effet, si le détecteur a_0 est bloqué à 1 à partir de l'état x_2 du diagnostiqueur D_A du vérin A (Figure 3-43) et que la fonction de prévision détectant ce défaut n'existe pas, alors l'état x_{14} ne pourrait pas retourner l'étiquette de défauts F1 au plus tôt. Il faudrait attendre le déplacement du vérin jusqu'à sa position de sortie a_1 pour que le diagnostiqueur voie le passage de a_1 de 0 à 1 comme un événement non attendu mais corrélé à l'ordre de sortie. Dès lors, le défaut ne doit pas être identifié sur a_1 mais bien comme un défaut sur a_0 qui est bloqué à 1, ce qui demande une analyse plus précise du procédé.

L'affectation d'un "1" sur chaque ligne permet de vérifier que tous les défauts peuvent être détectés au moins par un état du diagnostiqueur. Cette matrice ne représente pas une boucle d'incertitude puisque tout défaut est diagnosticable dans un retard borné.

Le diagnostiqueur D_A est capable de diagnostiquer tout défaut dans un retard borné (Figure 3-42). Le diagnostiqueur du vérin B, étant construit à l'identique, est également capable de diagnostiquer tout défaut dans un retard borné. Par conséquent, l'ensemble du système de transfert de pièces de la Figure 3-13 est co-diagnosticable pour l'ensemble des partitions que nous avons défini dans le Tableau 3-9.

3.6.3. Complexité

A partir des observations et du nombre d'éléments de PO constituant un procédé, il est possible d'établir la complexité des diagnostiqueurs. Cette complexité est établie uniquement pour la détection de défauts simples sans propagation. Elle est essentiellement liée au nombre de variables d'états composant les diagnostiqueurs. En effet, pour chaque état normal, il faut pouvoir exprimer tous les défauts diagnosticables sur chacun des composants (actionneurs et capteurs).

Proposition : Soit un diagnostiqueur local D_i, représentant les défauts observables et non observables, et d le nombre de variables d'états composant ce diagnostiqueur, alors la complexité de D_i est polynomiale et d'ordre $O_{Di}(d \times N)$. Si l'on simplifie le diagnostiqueur local D_i aux défauts non observables, alors celui-ci est de complexité $2N$.

Démonstration : A partir de chaque état normal d'un diagnostiqueur local D_i, il est possible d'évoluer au minimum vers un seul état normal. Dès lors, à partir de cet état normal, *d-1* défauts observables sont diagnosticables par l'observation des événements non attendus, atteignant chacun un état anormal, et un défaut non observable est diagnostiqué par au moins une fonction de prévision. Par conséquent, il existe pour chaque état normal, d états défaillants. Dès lors, le nombre d'états du diagnostiqueur est, au pire, égal à $(d+1)N$ où N représente le nombre d'états du fonctionnement normal. Simplifié aux défauts non observables, le diagnostiqueur ne comporte alors plus qu'un état défaillant pour un état

normal, soit par conséquent une complexité de (1+1)*N* états pour l'ensemble de ce diagnostiqueur simplifié.

Pour le diagnostiqueur D_A du vérin A (Figure 3-42), il y a $N = 8$ états normaux et $d = 4$ variables d'états (a_0, a_1, A+, A-). Par conséquent, la complexité du diagnostiqueur D_A est de $O_D = (4+1) \times 8 = 40$ états. Pour le diagnostiqueur D_A simplifié au défauts non observables (Figure 3-43), il n'y a plus que $(1+1) \times 8 = 16$ états.

La structure étant décentralisée, la complexité de l'ensemble des diagnostiqueurs locaux est égale à la somme de leur complexité.

3.6.4. Coordinateur et décision finale

Un diagnostiqueur local ne peut détecter que les défauts associés à ces éléments localement. Dès lors, lorsqu'un défaut ne peut pas être diagnostiqué par au moins un diagnostiqueur local, il faut pouvoir le diagnostiquer à un niveau plus élevé par l'intermédiaire de règles globales établies dans un coordinateur. En effet, les approches de diagnostic centralisées permettent de diagnostiquer l'ensemble des partitions de défauts à travers un diagnostiqueur global. Pour garantir les mêmes performances de diagnostic, les approches décentralisées doivent, par conséquent, pouvoir détecter l'ensemble de ces partitions par les diagnostiqueurs locaux et si besoin est par un coordinateur.

Les approches décentralisées impliquent également la gestion des interactions entre les différents éléments et cela notamment dans le cas où un détecteur se trouve partagé entre différents actionneurs et par conséquent par différents diagnostiqueurs. Cela peut créer des cas d'indécision sur l'occurrence d'un défaut en raison de l'observation partielle. Ces cas d'indécision doivent être résolus par le coordinateur.

Le coordinateur possède deux rôles essentiels : établir les contraintes de spécifications globales qui ne peuvent pas être définies par les diagnostiqueurs locaux et régler les ambiguïtés et les cas d'indécision entre les diagnostiqueurs locaux.

3.6.4.1. Etablissement des contraintes globales

Le premier rôle du coordinateur est d'analyser le modèle de la commande afin d'exprimer les spécifications globales du procédé qui ne peuvent pas être représentées localement. Il s'agit très souvent de comportements faisant appel à plusieurs éléments interagissant entre eux. Le coordinateur exprime des contraintes globales de sécurité et/ou de vivacité sur le procédé. Pour cela, nous avons choisi de modéliser ces contraintes par des règles logiques. Ces règles, devant être respectées à tout moment par le procédé, constituent un diagnostiqueur "d'expertise" de haut niveau dont la décision est prioritaire. La difficulté est de s'assurer que l'ensemble des contraintes globales, non décrites par des diagnostiqueurs locaux, soit exprimé.

Dans l'exemple du système de transfert de pièces présenté en Figure 3-13, une des contraintes globales du système concerne la sortie du vérin A qui ne peut s'effectuer que si le vérin B est en position b_0. En effet, si le vérin B est sorti et que le vérin A pousse une pièce, il y a un grand risque de détérioration de la pièce voire même de casse mécanique d'un vérin. Il faut donc pouvoir exprimer la contrainte "Ne pas sortir le vérin A si le vérin B n'est pas en b_0" retournant une étiquette de défaut de type F2 par une règle logique de type :

Règle 1	$\neg b_0 \wedge A+ = 0$	Etiquette F2

3.6.4.2. Gestion de l'ambiguïté et des cas d'indécision

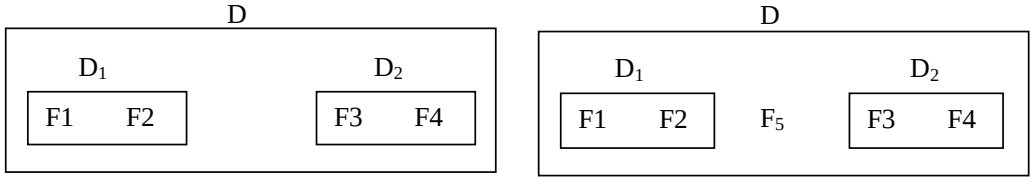
Un conflit décisionnel correspond à une prise de décision différente entre diagnostiqueurs locaux sur un élément qu'ils ont en commun. En effet, si deux diagnostiqueurs locaux D_i et D_j partagent en commun un détecteur k, alors la décision du diagnostiqueur local D_i sur l'état de fonctionnement du détecteur k doit être identique à la décision du diagnostiqueur local D_j. Dans le cas contraire, il y a un conflit sur la décision finale, conflit qui doit être résolu par le coordinateur par des règles de décision. Cette situation se présente donc lorsqu'une partition de défauts est détectable par au moins deux diagnostiqueurs locaux.

La Figure 3-44 présente trois cas pour 2 diagnostiqueurs locaux :

- La Figure 3-44a présente un cas simple où il n'y a pas de conflit entre les diagnostiqueurs D_1 et D_2. Il n'existe pas de partitions communes, donc il n'y a pas besoin de coordinateur puisqu'il n'y a pas de partitions de défauts non détectables par les diagnostiqueurs à un niveau local.

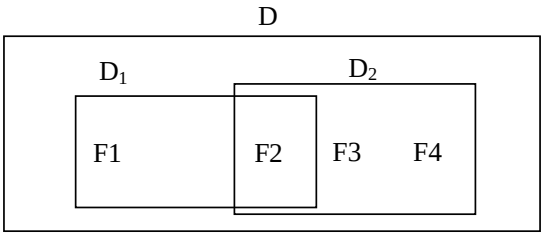
- La Figure 3-44b présente un cas sans conflit de décision entre les diagnostiqueurs D_1 et D_2 mais avec l'utilisation d'un coordinateur pour diagnostiquer les défauts de type F5 ne pouvant pas être détectés localement par les deux diagnostiqueurs.

- Enfin, la Figure 3-44c montre le cas d'un conflit décisionnel sur les défauts de type F2 qui sont diagnosticables à la fois par les diagnostiqueurs D_1 et D_2. Si ce défaut survient, il faut que les deux diagnostiqueurs retournent la même étiquette, sinon il y a un conflit. Pour lever ce risque de conflit, il faut faire appel à un coordinateur qui prendra la décision finale sur le fonctionnement du système.



a) Aucune partition en commun entre les diagnostiqueurs locaux

b) Défaut de type F5 diagnostiqué par le coordinateur



c) Défaut de type F2 en commun entre les diagnostiqueurs, risque d'un cas d'indécision

Figure 3-44 : Risque d'un cas d'indécision sur un défaut détecté par 2 diagnostiqueurs

3.6.4.3. Table des décisions

Les diagnostiqueurs locaux doivent échanger des messages sur la décision à prendre avec le coordinateur à travers un protocole de communication. Nous faisons l'hypothèse que cette communication est à délai nul et sans perte d'informations. Le protocole consiste à envoyer les étiquettes de décision locale de chaque diagnostiqueur D_i au coordinateur C qui les analyse à travers une table d'agrégation des décisions, dépendant du cahier des charges, permettant ainsi de répondre aux problèmes d'indécision.

Soient deux diagnostiqueurs D_1 et D_2 retournant une décision DD_1 et DD_2 et un coordinateur C représentant les contraintes globales à respecter et retournant une décision DC. On définit par l_1 la fonction de décision retournant l'ensemble des étiquettes de défauts de D_1, l_2 l'ensemble des étiquettes de défauts de D_2 et l_c l'ensemble des étiquettes de défauts de C. Les étiquettes possibles des diagnostiqueurs locaux sont Fi indiquant un défaut de type Fi, NFi garantissant l'absence du défaut de type Fi ou "–" représentant une indécision sur l'état de fonctionnement de l'élément du système entre un fonctionnement normal et défaillant. Les étiquettes possibles sur le coordinateur sont constituées par une étiquette Fj indiquant un défaut de type Fj et par une étiquette NFj garantissant l'absence du défaut de type Fj. Sous l'hypothèse qu'un diagnostic décentralisé puisse diagnostiquer l'ensemble des partitions de défauts {F1, F2, …, Fr}, il est possible d'agréger les décisions de l'approche de diagnostic décentralisée par :

- Si $l_1 \cap l_2 \neq \varnothing \Rightarrow$ il n'y a aucune partition en commun entre les diagnostiqueurs et la décision finale $DF = DD_1 + DD_2 + DC$.
- Si $l_1 \cap l_2 = F1 \Rightarrow$ la partition de défauts de type F1 est commune entre les diagnostiqueurs locaux avec un détecteur commun entre les deux éléments. Par conséquent la décision finale, DF, pour un défaut de type F1 est définie par le Tableau 3-12.

Cas	DD_1	DD_2	DC	Agrégation des décisions	DF
1	F1	F1		F1	F1
2	NF1	NF1		NF1	NF1
3	-	F1		F1	F1
4	-	NF1		NF1	NF1
5	-	-	F1	Indécision	F1
6	-	-	NF1	Indécision	NF1

Tableau 3-12 : Table des décisions sur un défaut de type F1

Les cas 1 et 2 montrent un accord entre les décisions locales des deux diagnostiqueurs locaux. Dès lors, la décision du coordinateur n'est donc pas utilisée et la décision finale, DF, est bien celle correspondante à la décision locale.

Les cas 3 et 4 indiquent une indécision sur le diagnostiqueur D_1, représenté par le symbole "-", alors que le diagnostiqueur D_2 prend une décision avec certitude. Dès lors, la décision finale revient à confirmer la décision du diagnostiqueur certain.

Enfin, les cas 5 et 6 montrent une ambiguïté de décision sur l'état des diagnostiqueurs D_1 et D_2. Le coordinateur doit intervenir pour prendre une décision finale soit par l'étiquette F1

pour un défaut, soit par l'étiquette NF1 pour garantir l'absence de ce défaut.

Il est ainsi possible de vérifier les performances du diagnostic décentralisé avec coordinateur par rapport à un diagnostic centralisé sur l'ensemble des partitions à diagnostiquer. La priorité à la décision du coordinateur est donnée à chaque ambiguïté de décision entre les diagnostiqueurs locaux.

Au final, l'agrégation des décisions de la table du coordinateur revient simplement à effectuer un opérateur logique de type OU entre l'ensemble des décisions locales et celle des règles du coordinateur (Figure 3-45). Dès qu'un défaut apparaît sur l'un des éléments, il doit être signalé à l'utilisateur.

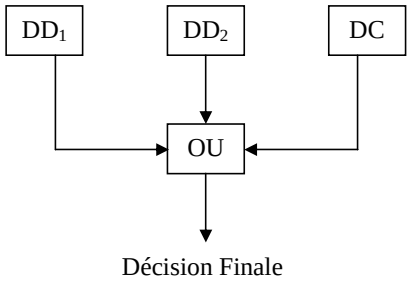


Figure 3-45 : Agrégation des décisions

3.6.4.4. Application à l'exemple

Nous allons regarder à nouveau l'exemple du système de transfert de pièces de la Figure 3-13. Pour que le coordinateur puisse gérer l'ensemble des cas d'indécision, il faut rappeler au préalable l'ensemble des partitions que les diagnostiqueurs locaux peuvent diagnostiquer. Le diagnostic décentralisé est composé de deux diagnostiqueurs dont les partitions de défauts sont rappelées en Tableau 3-13 et Tableau 3-14.

Π_{F1}	Défauts liés au détecteur a_0	Π_{F4}	Défauts liés au détecteur a_1
Π_{F2}	Erreur de commande sur l'ordre A+	Π_{F5}	Erreur de commande sur l'ordre A-
Π_{F3}	Défauts sur l'actionneur vérin A		

Tableau 3-13 : Partitions de défauts du diagnostiqueur D_A du vérin A

Π_{F6}	Défauts liés au détecteur b_0	Π_{F9}	Défauts liés au détecteur b_1
Π_{F7}	Erreur de commande sur l'ordre B+	Π_{F10}	Erreur de commande sur l'ordre B-
Π_{F8}	Défauts sur l'actionneur vérin B		

Tableau 3-14 : Partitions de défauts du diagnostiqueur D_B du vérin B

Chaque diagnostiqueur rend une décision locale DD_i correspondant à l'étiquette de défaut. Cependant, pour cet exemple, le diagnostiqueur D_A n'est pas capable de garantir l'absence de défauts dans les états x_1, x_2, x_3, x_4, x_5, x_6, x_7 et x_8 (Figure 3-42). Cela implique une étiquette d'indécision DD_A = "-". Le diagnostiqueur D_A permet uniquement de garantir avec certitude les états défaillants. Il en est de même pour le diagnostiqueur D_B.

Seul le coordinateur, possédant une règle de contrainte globale, est capable de garantir l'absence du défaut de type F2 lié à cette règle. Par conséquent, le coordinateur retourne

uniquement une étiquette de défaut F2 correspondant au non-respect de cette règle ou une étiquette NF2 garantissant la non-apparition de ce défaut.

La décision finale DF est prise par l'agrégation des différentes décisions des diagnostiqueurs locaux et du coordinateur. Le Tableau 3-15 présente cette décision finale avec $i \in \{1, 2,\ldots, 5\}$ et $j \in \{6, 7,\ldots, 10\}$.

Cas	DD_A	DD_B	DC	DF
1	-	-	NF2	NF2
2	Fi (i≠2)	-	NF2	Fi, NF2
3	-	Fj	NF2	Fj, NF2
4	Fi (i≠2)	Fj	NF2	FiFj, NF2
5	-	-	F2	F2
6	Fi (i≠2)	-	F2	Fi, F2
7	-	Fj	F2	Fj, F2
8	Fi (i≠2)	Fj	F2	Fi, Fj, F2
9	F2	-		F2
10	F2	Fj		F2, Fj

Tableau 3-15 : Table des décisions

Dès lors, si aucun défaut n'est diagnostiqué, les diagnostiqueurs locaux retournent une ambiguïté de décision et la décision finale revient à la décision du coordinateur : DF = DC = NF2 (cas 1). Dans le cas où un défaut est diagnostiqué sur un seul diagnostiqueur alors la décision finale DF retourne l'étiquette en défaut (cas 2 et 3). Le cas 5 montre un défaut non diagnosticable par aucun des deux diagnostiqueurs mais diagnosticable par la règle du coordinateur. Si un ou deux diagnostiqueurs et/ou le coordinateur sont en défauts, c'est un défaut multiple et la décision finale est l'ensemble des étiquettes de défauts (cas 4, 6, 7 et 8). Enfin, si le diagnostiqueur D_A retourne un défaut de type F2, alors il n'y a pas besoin de la décision de coordinateur (cas 9 et 10).

Comme il n'existe pas de détecteurs communs entre les diagnostiqueurs locaux, il n'y a donc pas besoin de vérifier le cas d'indécision sur l'occurrence d'un défaut appartenant à la partition en commun à ces deux diagnostiqueurs.

3.7. Conclusion

L'information contenue dans les Systèmes à Evénements Discrets (SED) est très limitée et difficilement interprétable directement. Il faut pouvoir faire ressortir à la fois l'aspect binaire, séquentiel et temporel d'un procédé à travers la structure de la Partie Opérative, les spécifications de la commande et le comportement temporel du système. L'efficacité d'un diagnostic des défaillances des SED, et plus particulièrement des systèmes manufacturiers, réside dans la cohabitation de toutes ces informations.

Nous avons ainsi opté pour une structure de diagnostic décentralisé avec coordinateur justifié par l'aspect décentralisé de l'information des systèmes manufacturiers. Cette décentralisation permet alors de diminuer le problème d'explosion combinatoire inhérent aux SED. Les modèles des composants élémentaires du système, appelés ici Eléments de Partie

Opérative EPO, sont caractérisés par des spécificités technologiques qui leur sont propres. Ils constituent une information de base primordiale à la réalisation d'un diagnostic puisqu'ils décrivent toutes les situations possibles et réalisables.

A partir de cette information, il faut pouvoir en extraire le comportement désiré de la commande spécifiée par GRAFCET ou par contraintes locales. La cohabitation de l'information de la PO et de la PC conduit à l'obtention d'Eléménts de Partie Opérative Commandés (EPOC). Sur chaque modèle décrivant le comportement désiré ($EPOC_i$), nous avons ensuite introduit une information temporelle par l'utilisation de fonctions de prévision pour obtenir des Eléménts de Partie Opérative Commandés Temporels (EPOCT). Nous avons ainsi réalisé la cohabitation de l'ensemble des informations présentes sur le système. Le modèle d'$EPOCT_i$ a été étendu pour prendre en compte l'ensemble des défauts diagnosticables. Il s'agit de générer un super état de défaut qui va être activé suite à l'apparition soit d'un événement qui n'est pas celui attendu par le modèle, soit par la violation d'une fonction de prévision. La localisation des défauts est réalisée par la construction de diagnostiqueurs locaux où chaque état est associé à une fonction de décision.

Pour vérifier les capacités de diagnostic des différents diagnostiqueurs, nous avons établi une notion de co-diagnosticabilité mixte à base d'événements, à base d'états et à base d'informations temporelles. Cette notion vérifie notamment qu'il est possible de diagnostiquer l'ensemble des partitions définies dans un délai borné. Afin d'exprimer les différentes réactivités entre les EPO et de résoudre les cas d'indécision entre diagnostiqueurs locaux, nous avons mis en place un coordinateur. Ce coordinateur établit les spécifications globales qui n'ont pu être exprimées par au moins un diagnostiqueur local. Il est également responsable de la décision finale à retourner à l'utilisateur sur l'état du procédé par agrégation des décisions locales.

Afin de montrer l'applicabilité de notre approche de diagnostic décentralisée des SED pour les applications manufacturières, nous allons présenter, dans le chapitre 4, deux exemples. Le premier exemple est simple et se focalise sur le risque d'explosion combinatoire inhérent aux SED. Le second exemple est un système manufacturier plus complexe, composé d'éléments de PO de technologie différente. L'objectif de cet exemple est de montrer les différentes étapes d'obtention des diagnostiqueurs locaux, du coordinateur et de la prise de décision finale.

Chapitre 4 : Application au diagnostic des Systèmes à Evénements Discrets

4.1. Introduction

Nous avons vu dans le chapitre précédent comment obtenir un diagnostic décentralisé avec coordinateur pour les applications manufacturières. Ce diagnostic est basé sur une modélisation modulaire des composants d'un procédé autour d'automates à états enrichis. La construction d'un diagnostiqueur local s'effectue à partir de modèles pratiques de la PO validés par un expert. Ces modèles sont enrichis par l'intégration d'une commande afin de décrire localement le comportement désiré de chaque composant. Enfin, une information temporelle a été introduite afin de représenter les contraintes temporelles à appliquer sur les composants du système. Cette décentralisation des diagnostiqueurs implique de gérer les interactions globales du procédé. Pour cela, nous avons fait appel à un coordinateur permettant au final de garantir des performances de diagnostic égales à une structure de diagnostic centralisée.

Afin de montrer les avantages et les performances de l'approche proposée dans le chapitre 3, nous proposons dans ce chapitre deux applications. Dans un premier temps, nous allons regarder l'intérêt d'une structure décentralisée afin de diminuer le risque d'explosion combinatoire sur les diagnostiqueurs autour d'un exemple simple de wagonnet. Nous étudierons ensuite un système manufacturier plus complexe de tri de caisses composé de différents actionneurs permettant à la fois de justifier l'aspect décentralisé du diagnostic, mais également de démontrer sa performance. Enfin, nous présentons un outil de simulation basé sur *Stateflow* de Matlab que nous avons développé pour montrer l'applicabilité de l'approche.

4.2. Exemple 1 : Système de wagonnet

Nous allons procéder, dans ce premier exemple, à une comparaison entre une structure centralisée et une structure décentralisée. L'objectif est de montrer exclusivement la différence en terme de nombre d'états. Nous réalisons cette comparaison uniquement sur les modèles de PO Commandés intégrés aux diagnostiqueurs sans information temporelle. En effet, l'information temporelle n'intervient pas dans l'augmentation du nombre d'états puisqu'elle est représentée par des fonctions de prévision introduites dans chaque état.

4.2.1. Présentation de l'exemple

L'exemple concerne un wagonnet composé d'un bouton Marche/Arrêt, d'un chariot et d'un vérin [PHI 05a], [PHI 05d]. Le chariot se déplace latéralement grâce à un moteur

commandé par les ordres *G* pour aller à gauche et *D* pour aller à droite avec trois capteurs Tout Ou Rien *a*, *b* et *c* permettant d'identifier la position du chariot. Pour évacuer le contenu du chariot, le vérin est un double effet commandé en sortie par l'ordre *SO* et en rentrée par *RE* avec deux capteurs fin de course, *fcr* quand il est rentré et *fcs* quand il est sorti. La Figure 4-1 illustre le système.

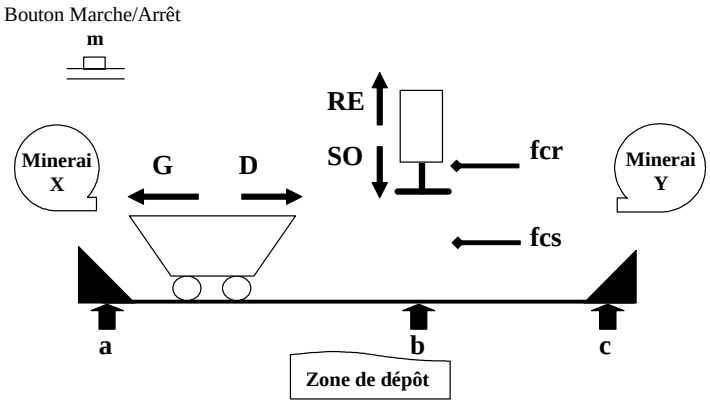


Figure 4-1 : Système de wagonnet

L'objectif de cette installation est de transporter un mélange de minerais X et Y dans une zone de dépôt. L'appui sur le bouton Marche/Arrêt ($m = 1$) lance le cycle. En position initiale, le chariot se trouve en *a* pour être chargé avec du minerai de type X. Il se dirige ensuite vers la position *c* pour compléter son contenu avec du minerai Y. Le chariot doit finalement retourner en position *b* pour décharger le tout. Le vérin doit alors être commandé en sortie afin de faire basculer le contenu du chariot dans la zone de dépôt. La rentrée du vérin remet le chariot vide sur ses rails. Le chariot retourne ensuite en position initiale pour un nouveau cycle si le bouton est toujours en position marche $m = 1$. Si *m* est invalidé, alors le système termine son cycle et revient en position initiale pour attendre une nouvelle mise en marche.

Pour cette étude de cas, les hypothèses de travail suivantes sont émises :

- Il n'y a aucune ressource commune. Chaque actionneur est associé à ses propres capteurs et chaque capteur n'est utilisé que par un seul actionneur,
- Les capteurs réagissent de manière instantanée,
- L'inertie du chariot est considérée comme nulle,
- Le système effectue toujours le même cycle de fonctionnement. Il ne présente donc pas d'évolutions parallèles ou complexes,
- On ne considère qu'un seul défaut possible sur chaque élément de PO. Par conséquent, si l'utilisateur observe deux défauts sur son système, alors ces deux défauts ne proviennent forcément pas du même élément de PO.
- Aucun défaut n'est possible sur le bouton Marche/Arrêt puisqu'il n'est pas considéré comme un élément de PO.

Nous lèverons une partie de ces hypothèses dans le paragraphe 4.3 par un second exemple.

4.2.2. Les différentes étapes de l'approche

Que ce soit pour une structure de diagnostic centralisée ou décentralisée, l'approche d'obtention d'un diagnostiqueur global (structure centralisée) ou des diagnostiqueurs locaux (structure décentralisée) consiste à :

1) Recueillir la commande appliquée sur le système par la construction du Graphe Equivalent *GE* au GRAFCET de commande,
2) Construire les modèles de PO de chaque élément du système (EPO_i), en choisissant les modèles des détecteurs et actionneurs présents dans la bibliothèque,
3) Etablir l'ensemble des partitions de défauts à diagnostiquer,

Pour la structure centralisée :

4) Etablir le modèle global de PO par composition asynchrone de tous les EPO,
5) Appliquer l'algorithme 1 ou 2 d'intersection, donné au paragraphe 3.3.1., entre le *GE* de la commande et le modèle global de la PO pour obtenir le comportement désiré global POCG,
6) Intégrer l'information temporelle à travers les fonctions de prévision dans chaque état de la PO Commandée Globale (POCG) afin d'obtenir la POCG Temporelle (POCGT),
7) Etablir le diagnostiqueur global, D_G, pour la POCGT.

Pour la structure décentralisée :

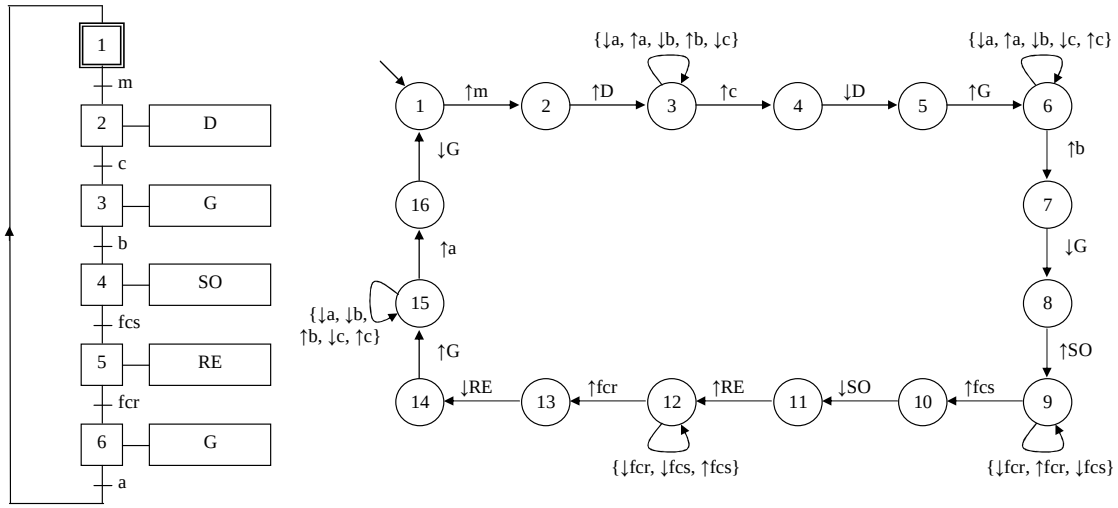
4) Appliquer l'algorithme 1 ou 2 d'intersection entre le *GE* de la commande et chaque EPO_i pour obtenir le comportement désiré localement $EPOC_i$,
5) Intégrer l'information temporelle à travers les fonctions de prévision dans chaque état des EPO Commandés ($EPOC_i$) pour les $EPOC_i$ Temporels ($EPOCT_i$),
6) Etablir les diagnostiqueurs locaux, D_i, pour chaque $EPOCT_i$,
7) Construire le coordinateur établissant les contraintes globales du système et gérant les cas d'indécision entre les diagnostiqueurs locaux afin de pouvoir diagnostiquer tous les défauts que le diagnostiqueur global est capable de diagnostiquer.

4.2.3. Constitution du Graphe Equivalent GE

La première étape de l'approche consiste à obtenir le Graphe Equivalent (*GE*). Ce *GE* est commun aux deux structures. La commande est présentée Figure 4-2a sous forme GRAFCET. De l'étape initiale, il faut attendre l'appui sur *m* avant d'envoyer l'ordre droite *D* en étape 2 jusqu'en position *c*. Dès lors, dans l'étape 3, l'ordre gauche *G* est envoyé jusqu'en *b* où le vérin effectue ensuite un aller-retour pour décharger le chariot (étapes 4 et 5). Lorsque le vérin est revenu en *fcr*, le chariot retourne en position initiale *a* après l'envoi de l'ordre gauche *G* à l'étape 6.

De ce modèle, nous allons ensuite extraire un Graphe Equivalent *GE* (Figure 4-2b), automate événementiel à 16 états. De l'état 1, il faut attendre l'activation de *m* pour passer à

l'état 2. L'activation de l'ordre *D* correspondant à l'étape 2 du GRAFCET permet d'aller vers l'état 3. Cet ordre engendre différentes réactions sur la PO représentées par une boucle d'événements non commandables associés à cet ordre sur l'état 3. C'est l'activation du détecteur *c* qui va permettre de quitter l'état 3 vers l'état 4 correspondant à l'étape 3 du GRAFCET. Cette étape engendre tout d'abord la désactivation de l'ordre *D* de l'étape précédente (état 5 du *GE*), puis l'activation de l'ordre *G* (état 6) où des réactions sur la PO sont modélisées par une boucle d'événements. L'automate *GE* est construit selon l'algorithme proposé dans le paragraphe 3.3.1 du chapitre 3.



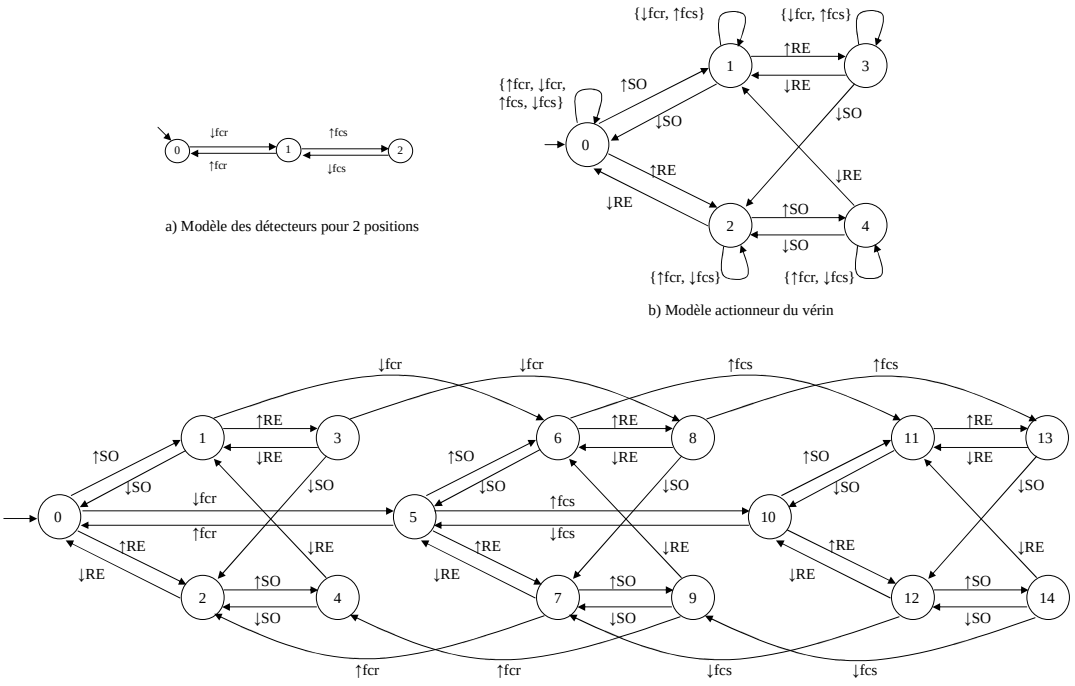
a) GRAFCET du système de wagonnet

b) GE du GRAFCET du système de wagonnet

Figure 4-2 : GRAFCET et Graphe Equivalent GE du wagonnet de la Figure 4-1

4.2.4. Modèles des Eléments de Partie Opérative EPO

La partie opérative est constituée de deux actionneurs : un vérin double effet piloté par un distributeur 5/2 bistable et un chariot. Nous rappelons pour le vérin le modèle des détecteurs en Figure 4-3 et celui de l'actionneur en Figure 4-3b. Après composition synchrone entre le modèle des détecteurs et le modèle actionneur, le modèle pratique final EPO_1 du vérin est présenté en Figure 4-3c et comprend 15 états. Le second actionneur est le chariot qui correspond à un moteur à deux sens de rotation. La construction du modèle EPO_2 du chariot est rappelée en Figure 4-4. Le modèle pratique final EPO_2 du chariot est un automate à 30 états ayant une structure "actionneur" répétée par le nombre d'états du modèle des détecteurs. Pour une question de lisibilité, nous ne détaillons pas le modèle EPO_2 du chariot lorsque la structure "actionneur" à 5 états est répétée plusieurs fois (Figure 4-4c).

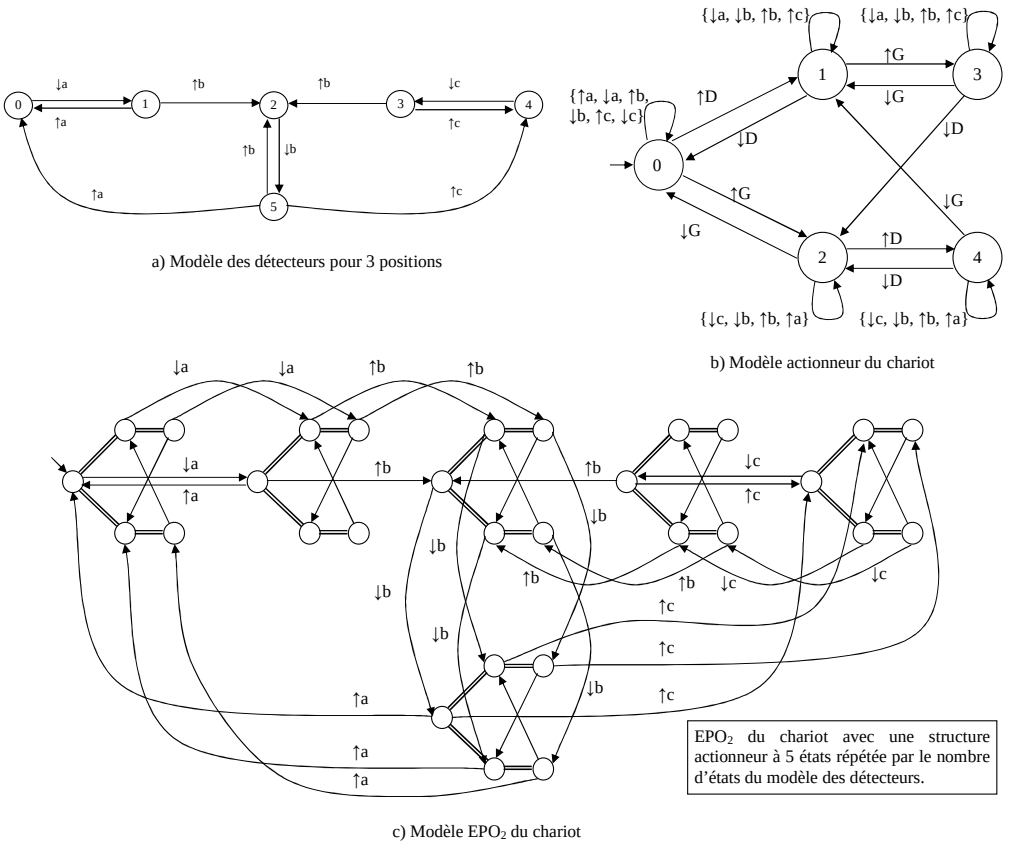


a) Modèle des détecteurs pour 2 positions

b) Modèle actionneur du vérin

c) Modèle EPO_1 du vérin

Figure 4-3 : Construction du modèle EPO_1 du vérin



a) Modèle des détecteurs pour 3 positions

b) Modèle actionneur du chariot

c) Modèle EPO_2 du chariot

Figure 4-4 : Construction de l'Elément de Partie Opérative EPO_2 du chariot

4.2.5. Partitions de défauts

A partir des EPO, il faut établir l'ensemble des défauts à diagnostiquer du procédé (étape 3 de l'approche). Pour cela, nous avons classé les défauts dans 11 partitions correspondant au Tableau 4-1 et définies ainsi :

- Les défauts sur le détecteur *a* sont dans la partition $\Pi_{F1} = \{f_1, f_2, f_3, f_4\}$ et leur diagnostic est indiqué par l'étiquette F1,
- les défauts sur le détecteur *b* sont dans la partition $\Pi_{F2} = \{f_5, f_6, f_7, f_8\}$ et leur diagnostic est indiqué par l'étiquette F2,
- les défauts sur le détecteur *c* sont dans la partition $\Pi_{F3} = \{f_9, f_{10}, f_{11}, f_{12}\}$ et leur diagnostic est indiqué par l'étiquette F3,
- les défauts liés au déplacement du chariot vers la droite (sur une erreur de la PC) sont dans la partition $\Pi_{F4} = \{f_{13}, f_{14}, f_{15}, f_{16}\}$ avec l'étiquette F4,
- les défauts liés au déplacement du chariot vers la gauche (sur une erreur de la PC) sont dans la partition $\Pi_{F5} = \{f_{17}, f_{18}, f_{19}, f_{20}\}$ avec l'étiquette F5,
- les défauts sur le détecteur *fcr* sont dans la partition $\Pi_{F6} = \{f_{21}, f_{22}, f_{23}, f_{24}\}$ et leur diagnostic est indiqué par l'étiquette F6,
- les défauts sur le détecteur *fcs* sont dans la partition $\Pi_{F7} = \{f_{25}, f_{26}, f_{27}, f_{28}\}$ et leur diagnostic est indiqué par l'étiquette F7,
- les défauts liés à la sortie du vérin (sur une erreur de la PC) sont dans la partition $\Pi_{F8} = \{f_{29}, f_{30}, f_{31}, f_{32}\}$ avec l'étiquette F8,
- les défauts liés à la rentrée du vérin (sur une erreur de la PC) sont dans la partition $\Pi_{F9} = \{f_{33}, f_{34}, f_{35}, f_{36}\}$ avec l'étiquette F9,
- les défauts liés au chariot qui reste bloqué dans une situation fixe sont dans la partition $\Pi_{F10} = \{f_{37}, f_{38}, f_{39}\}$ avec l'étiquette F10,
- les défauts liés au vérin qui reste bloqué dans une situation fixe sont dans la partition $\Pi_{F11} = \{f_{40}, f_{41}\}$ avec l'étiquette F11.

F1	f_1	Passage inattendu de a de 0 à 1	F6	f_{21}	Passage inattendu de fcr de 0 à 1
	f_2	Détecteur a bloqué à 0		f_{22}	Détecteur fcr bloqué à 0
	f_3	Passage inattendu de a de 1 à 0		f_{23}	Passage inattendu de fcr de 1 à 0
	f_4	Détecteur a bloqué à 1		f_{24}	Détecteur fcr bloqué à 1
F2	f_5	Passage inattendu de b de 0 à 1	F7	f_{25}	Passage inattendu de fcs de 0 à 1
	f_6	Détecteur b bloqué à 0		f_{26}	Détecteur fcs bloqué à 0
	f_7	Passage inattendu de b de 1 à 0		f_{27}	Passage inattendu de fcs de 1 à 0
	f_8	Détecteur b bloqué à 1		f_{28}	Détecteur fcs bloqué à 1
F3	f_9	Passage inattendu de c de 0 à 1	F8	f_{29}	Passage inattendu de SO de 0 à 1
	f_{10}	Détecteur c bloqué à 0		f_{30}	Activation de SO non envoyée
	f_{11}	Passage inattendu de c de 1 à 0		f_{31}	Passage inattendu de SO de 1 à 0
	f_{12}	Détecteur c bloqué à 1		f_{32}	Désactivation de SO non envoyée
F4	f_{13}	Passage inattendu de D de 0 à 1	F9	f_{33}	Passage inattendu de RE de 0 à 1
	f_{14}	Activation de D non envoyée		f_{34}	Activation de RE non envoyée
	f_{15}	Passage inattendu de D de 1 à 0		f_{35}	Passage inattendu de RE de 1 à 0
	f_{16}	Désactivation de D non envoyée		f_{36}	Désactivation de RE non envoyée
F5	f_{17}	Passage inattendu de G de 0 à 1	F10	f_{37}	Chariot bloqué en position a
	f_{18}	Activation de G non envoyée		f_{38}	Chariot bloqué en position b
	f_{19}	Passage inattendu de G de 1 à 0		f_{39}	Chariot bloqué en position c
	f_{20}	Désactivation de G non envoyée	F11	f_{40}	Vérin bloqué en position rentrée
				f_{41}	Vérin bloqué en position sortie

Tableau 4-1 : Ensemble des défauts à diagnostiquer pour l'exemple de la Figure 4-1

4.2.6. Structure centralisée

4.2.6.1. Partie Opérative Globale

La structure centralisée est basée sur un modèle global du procédé et un seul diagnostiqueur. Il faut constituer le modèle global du procédé obtenu par composition synchrone des différents modèles d'EPO. Les événements de l'EPO_1 du vérin étant différents des événements de l'EPO_2 du chariot, la composition asynchrone de ces deux modèles va conduire à un modèle global de la PO comprenant 15x30 = 450 états et 2520 transitions. Pour deux actionneurs, le modèle global est difficilement représentable.

4.2.6.2. Partie Opérative Globale Commandée

L'étape suivante consiste à effectuer la composition synchrone entre le modèle global de la PO et l'automate du *GE* de la Figure 4-2b. Il en résulte un automate à 23 états présenté Figure 4-5. Cet automate décrit le comportement désiré de la commande à travers tous ses événements. Il suffit alors de restreindre cet automate au langage que l'on souhaite diagnostiquer. Dans le cadre de cet exemple, nous souhaitons diagnostiquer tous types de défauts sur le vérin et sur le chariot. Le fonctionnement du bouton n'est pas considéré comme élément de la PO et ne fait donc pas partie du langage désiré. Tous les états reliés entre eux par un événement relatif à ce bouton sont donc agrégés en un groupe. C'est le cas des états 1 et 2 qui ne représentent plus qu'un seul et même état après agrégation. Au final, nous obtenons un modèle de PO Commandée Globale POCG à 22 états (Figure 4-6).

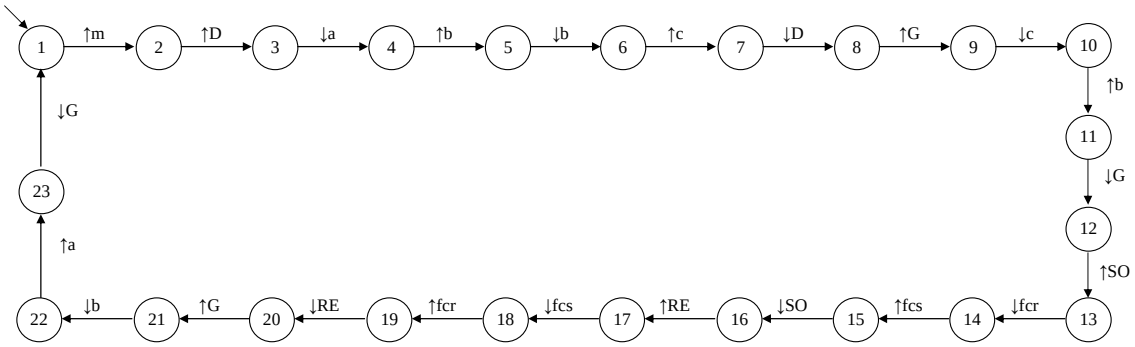


Figure 4-5 : Composition synchrone entre la PO Globale et l'automate équivalent GE

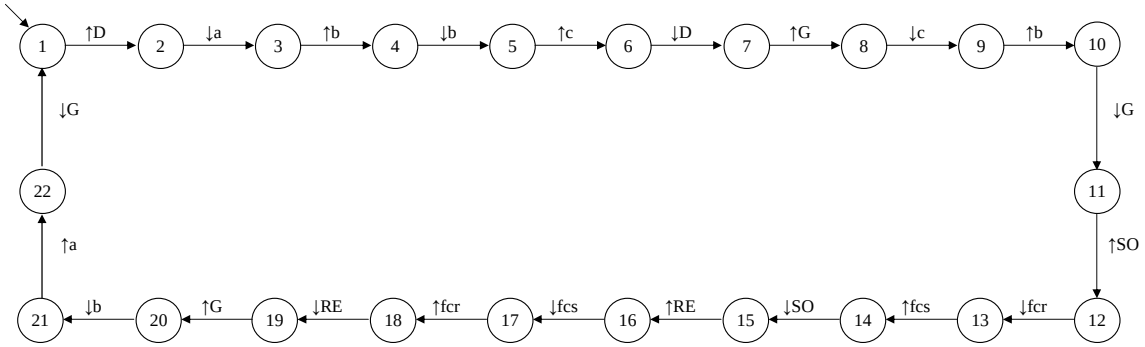


Figure 4-6 : Modèle de PO Commandée Globale POCG

4.2.6.3. Intégration de l'information temporelle

L'information temporelle, modélisée par les fonctions de prévision, est ajoutée au modèle de POCG afin d'obtenir la POCG Temporelle (POCGT). Chaque état possède alors une fonction de prévision du type $FP_x = FP(\alpha_1, \alpha_2) = \{\alpha_1, x, (\alpha_2, [t_{min}, t_{max}], x', l_x)\}$ enclenchée par l'événement entrant α_1 dans cet état et qui doit être respectée.

Par exemple à partir de l'état 2 de la POCG de la Figure 4-6, une fonction de prévision est définie dans la POCGT par $FP_{x2} = FP(\uparrow D, \downarrow a) = \{\uparrow D, x_2, (\downarrow a, [1s, 2s], x_3, \{F1, F10\})\}$ où l'événement ↓a doit apparaître dans un intervalle [1s, 2s] après l'occurrence de l'événement ↑D. Dans le cas contraire, la fonction de prévision indique un défaut concernant :

- Soit le détecteur *a* qui reste bloqué à 1 malgré le déplacement du chariot. Il s'agit donc

d'un défaut f_4 appartenant à la partition d'étiquette F1,

- Soit un défaut f_{37} correspondant au chariot bloqué en position a. Ce défaut appartient à la partition Π_{F10}.

Pour une question de lisibilité et pour appuyer davantage sur l'aspect explosion combinatoire, nous n'énumérerons pas l'ensemble des fonctions de prévision dans cet exemple.

4.2.6.4. Diagnostiqueur global

La cohabitation des informations provenant de la PO, de la PC et des contraintes temporelles dans un automate enrichi permet de constituer la base du diagnostiqueur global. Avec les différents défauts présentés dans le Tableau 4-1, il est possible de constituer un diagnostiqueur global du procédé où chaque état dispose d'une étiquette indiquant le comportement normal ou défaillant du procédé. A chaque état normal, il est possible de détecter un défaut de chaque partition.

Comme le vecteur d'état $V_1(x) = (a, b, c, D, G, fcr, fcs, SO, RE)$ contient 9 variables d'états, il y a donc la possibilité d'avoir 9 événements. Un de ces événements est attendu et correspond à un fonctionnement normal, tandis que les 8 autres événements correspondent à un fonctionnement anormal et conduisent vers des états défaillants. Chacun de ces états défaillants possède une étiquette indiquant la partition de défauts conduisant à cet état. De plus, il existe au moins une fonction de prévision permettant de détecter un défaut non observable par rapport à l'événement attendu. Il y a donc un minimum de 9 défauts possibles à diagnostiquer à partir de chaque état normal.

Par conséquent, le diagnostiqueur global final possède 22 états normaux et $22 \times 9 = 198$ états de défauts soit un total de 220 états. La Figure 4-7 représente un extrait du diagnostiqueur global simplifié aux défauts non observables composé de 44 états. A partir de l'état x_2 avec la fonction de prévision $FP_{x2} = FP(\uparrow D, \downarrow a) = \{\uparrow D, x_2, (\downarrow a, [1s, 2s], x_3, F1)\}$, il est possible d'évoluer vers l'état x_3 en situation normale ou d'évoluer vers un état de défaut soit par le non respect d'au moins une fonction de prévision associée à l'état x_{31}, $FP_{x2} = 1$, soit par l'observation d'un événement non attendu.

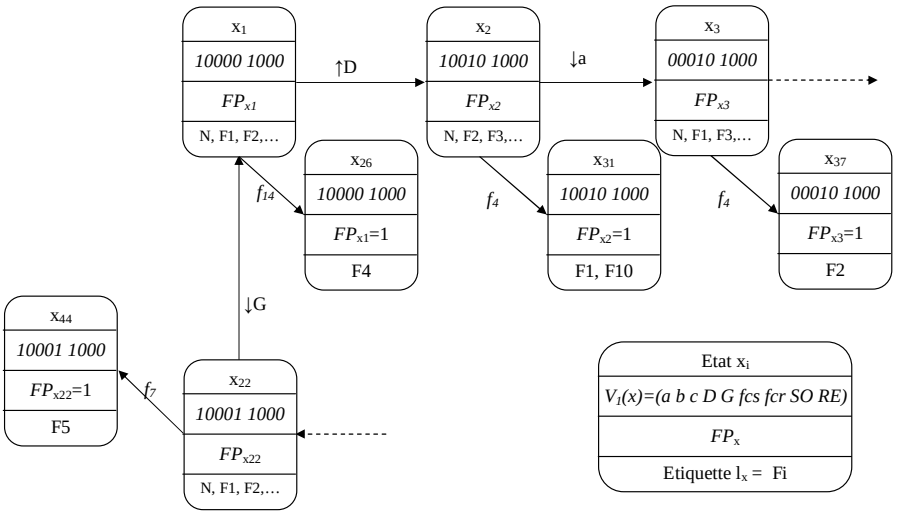


Figure 4-7 : Extrait du diagnostiqueur global

4.2.7. Structure décentralisée

L'approche de diagnostic avec structure décentralisée consiste, à partir de chaque modèle d'EPO_i, à obtenir des diagnostiqueurs locaux. Dans l'exemple du wagonnet de la Figure 4-1, il y a deux modèles d'EPO ce qui implique deux diagnostiqueurs. Il faut également définir les partitions de défauts diagnosticables par chaque diagnostiqueur. Pour cela, nous reprenons le Tableau 4-1 référençant l'ensemble des défauts et nous affectons les partitions de défauts Π_{F1}, Π_{F2}, Π_{F3}, Π_{F4}, Π_{F5} et Π_{F10} à l'EPO_2 du chariot et Π_{F6}, Π_{F7}, Π_{F8}, Π_{F9} et Π_{F11} à l'EPO_1 du vérin.

Chaque diagnostiqueur est obtenu par l'étape d'intersection entre la commande et l'EPO_i concerné afin d'obtenir le comportement désiré local de chaque EPO_i. Nous avons vu dans le chapitre 3 deux algorithmes d'intersection. L'algorithme 1 consiste à effectuer, dans un premier temps, la composition synchrone entre l'automate de la commande *GE* et chaque EPO, puis dans un second temps à réaliser une restriction au langage de l'EPO_i concerné. L'algorithme 2, quant à lui, restreint tout d'abord le langage de l'EPO_i concerné sur l'automate *GE*, puis effectue la composition synchrone avec l'EPO_i. Au final, les deux algorithmes donnent le même résultat, c'est-à-dire des EPO Commandés (EPOC). Nous allons voir dans ce paragraphe l'influence de ces deux algorithmes sur le nombre d'états.

4.2.7.1. Utilisation de l'algorithme 1 d'intersection

La première étape de l'algorithme 1 d'intersection consiste à faire la composition synchrone de la commande avec les EPO. Pour le vérin, la composition du modèle *GE* (Figure 4-2b) avec le modèle EPO_1 (Figure 4-3) donne un automate $GEPO_1$ à 18 états représenté en Figure 4-8. Cet automate présente le comportement du vérin et inclut des boucles d'événements non commandables correspondant au vérin. Par exemple, l'état 9 du GE de la Figure 4-2b qui présente tous les événements non commandables du vérin se trouve développé par les événements autorisés par l'EPO_1 de la Figure 4-3. Dès lors, à partir de l'état 9 du $GEPO_1$, seule la désactivation du détecteur *fcr* est possible avant l'activation de *fcs*. Par contre, tous les événements non commandables référant au modèle du chariot ne sont pas développés et sont toujours présents par des boucles d'événements (états 3, 6 et 17).

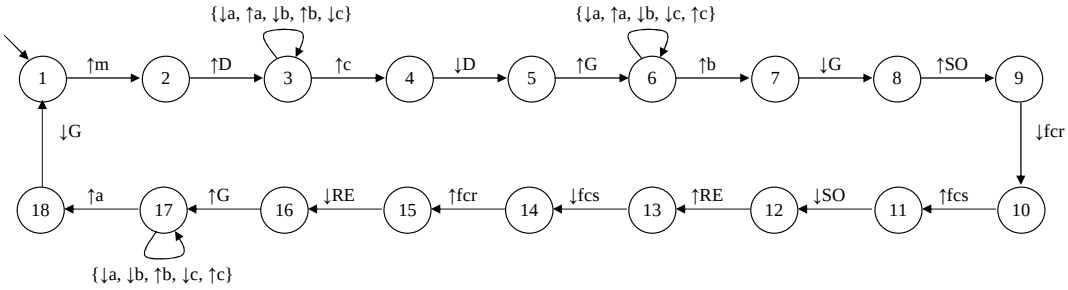


Figure 4-8 : Composition synchrone des automates GE et EPO_1 du vérin : $GEPO_1$

La seconde étape est la restriction du langage concerné. Pour le vérin, l'ensemble des événements du langage est {↑fcr, ↓fcr, ↑fcs, ↓fcs, ↑SO, ↓SO, ↑RE, ↓RE}. Tout autre événement reliant deux états est considéré comme non observable et entraîne l'agrégation des états en un groupe. A partir de l'état initial de l'automate $GEPO_1$, les états 1, 2, 3, 4, 5, 6, 7 et 8 sont reliés entre eux par des événements n'appartenant pas au vérin, ils constituent donc un groupe Gr1. Par contre, les états 9, 10, 11, 12, 13, 14 et 15 sont tous reliés par un événement appartenant au langage du vérin. Chacun d'eux constitue un groupe (Gr2, Gr3, Gr4, Gr5, Gr6,

Gr7 et Gr8). Les états 16, 17, 18, 1, 2, 3, 4, 5, 6, 7 et 8 sont agrégés dans un groupe Gr9. Ce groupe comprend également le groupe Gr1. A eux deux, ils constituent le même groupe. Au final, l'automate obtenu à partir des 8 groupes de l'automate $GEPO_1$ est celui de la Figure 4-9 composé de 8 états et décrivant l'Elément de PO Commandé du vérin $EPOC_1$.

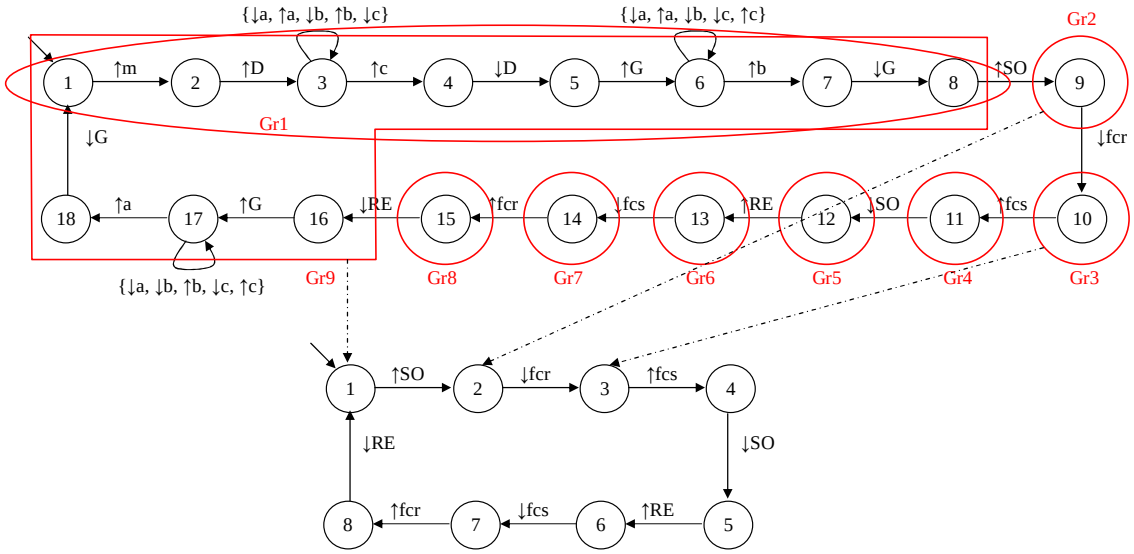


Figure 4-9 : Elément de PO Commandé $EPOC_1$ du vérin $EPOC_1$

Pour le chariot, la méthode est identique avec tout d'abord la composition synchrone entre le modèle *GE* de la Figure 4-2b et l'EPO_2 du chariot de la Figure 4-4c. Il en résulte l'automate $GEPO_2$ représenté en Figure 4-10 avec 21 états détaillant le fonctionnement du chariot par rapport à la commande désirée. De même que pour le vérin, il faut définir ensuite l'ensemble des événements du langage associé au chariot {↑a, ↓a, ↑b, ↓b, ↑c, ↓c, ↑D, ↓D, ↑G, ↓G} pour réaliser la restriction du langage sur l'automate $GEPO_2$. Après agrégation des états, il en résulte un modèle d'EPO Commandé du chariot $EPOC_2$ représenté en Figure 4-11 composé de 14 états décrivant uniquement le fonctionnement du chariot.

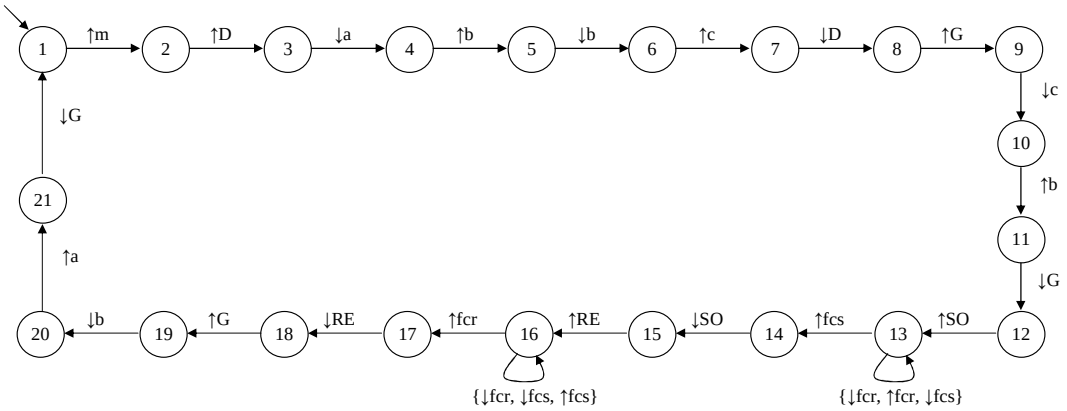


Figure 4-10 : Composition synchrone des automates GE et EPO_2 du chariot : $GEPO_2$

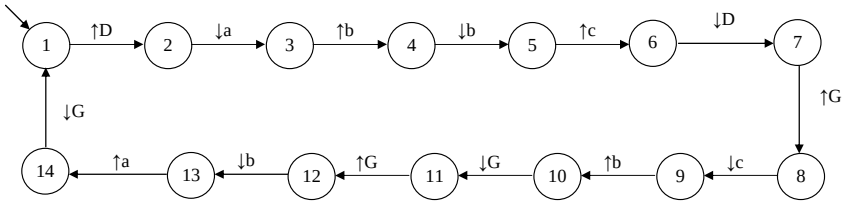


Figure 4-11 : Elément de PO Commandé $EPOC_2$ du chariot

4.2.7.2. Utilisation de l'algorithme 2 d'intersection

L'algorithme 2 d'intersection commence par restreindre le langage désiré sur le *GE* de la commande. Pour le vérin, l'ensemble des événements du langage est {↑fcr, ↓fcr, ↑fcs, ↓fcs, ↑SO, ↓SO, ↑RE, ↓RE}. Dès lors, les états 1, 2, 3, 4, 5, 6, 7, 8, 14, 15 et 16 reliés par des événements n'appartenant pas au vérin, constituent un groupe. Après restriction et agrégation des états, il en résulte un automate GE Restreint au langage de l'EPO_1 du vérin $GEREPO_1$ composé uniquement de 6 états (Figure 4-12). Cet automate exprime la commande désirée de l'EPO_1 du vérin mais ne détaille pas son fonctionnement. C'est pourquoi, il existe encore la présence des boucles d'événements correspondant au vérin sur les états 2 et 5. C'est la composition synchrone de cet automate avec l'EPO_1 (Figure 4-3) qui donne le fonctionnement et conduit au même modèle d'$EPOC_1$ que celui de la Figure 4-9.

Pour le chariot, il faut appliquer le même principe en restreignant tout d'abord le langage du *GE* conduisant à un automate GE Restreint au langage de l'EPO_2 $GEREPO_2$ à 9 états (Figure 4-13). La composition synchrone de cet automate avec celui de l'EPO_2 de la Figure 4-4 donne au final le même automate d'$EPOC_2$ que celui de la Figure 4-11.

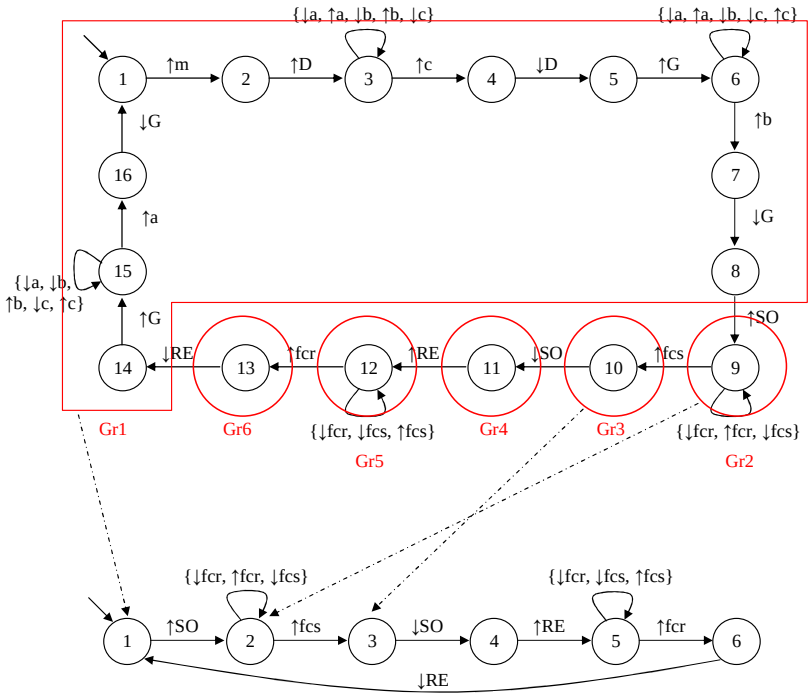


Figure 4-12 : GE restreint au langage du vérin : $GEREPO_1$

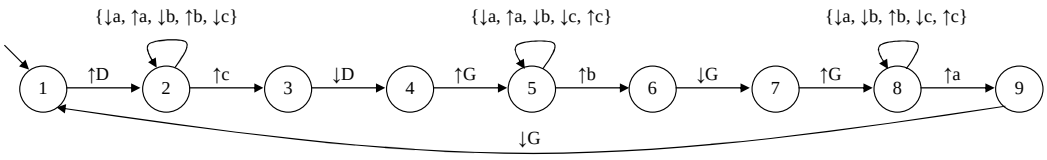


Figure 4-13 : GE restreint au langage du chariot : $GEREPO_2$

L'information temporelle est identique à celle d'une structure centralisée où les fonctions de prévision sont affectées à chaque état pour chacun des deux $EPOC_1$ et $EPOC_2$. Nous supposons que ces contraintes temporelles sont définies entre les événements observables pour chacun des EPOC Temporels (EPOCT).

4.2.7.3. Diagnostiqueurs locaux

La cohabitation d'informations provenant de la PO, de la PC et des réactivités temporelles des actionneurs avec la prise en compte des différents défauts par l'affectation d'étiquettes à un niveau local, permet d'aboutir à deux diagnostiqueurs locaux.

Pour le vérin, le diagnostiqueur local D_1 est établi à partir des 8 états normaux de l'$EPOCT_1$ et de l'ensemble des défauts qu'il est possible de détecter à partir de chaque état. A partir de chaque état du comportement désiré, tout événement ne correspondant pas à l'événement attendu ou bien encore le passage d'une des fonctions de prévision à 1, conduit à un état défaillant. Il y a donc 4 états défaillants, au maximum, pour chaque état normal qui aboutit au final à $8 \times 4 = 32$ états défaillants auxquels viennent s'ajouter 8 états du comportement normal. Soit au total 40 états qui forment le diagnostiqueur local, D_1, du vérin.

La Figure 4-14 présente un extrait du diagnostiqueur local du vérin D_1 où, à partir de l'état x_1, il est possible de diagnostiquer un défaut de type F8 par la fonction de prévision $FP_{x1} = FP(\downarrow RE, \uparrow SO) = \{\downarrow RE, x_1, (\uparrow SO, [1s, 2s], x_2, F8)\}$ ou un défaut de type F6, F7 ou F9 par l'occurrence d'un événement de défaut et le vecteur d'état $V_1(x) = (fcr\ fcs\ SO\ RE)$. Les défauts observables étant détectables très facilement par l'opérateur logique "OU exclusif", la Figure 4-15 représente le diagnostiqueur D_1 simplifié aux défauts non observables qui peuvent être détectés par des fonctions de prévision.

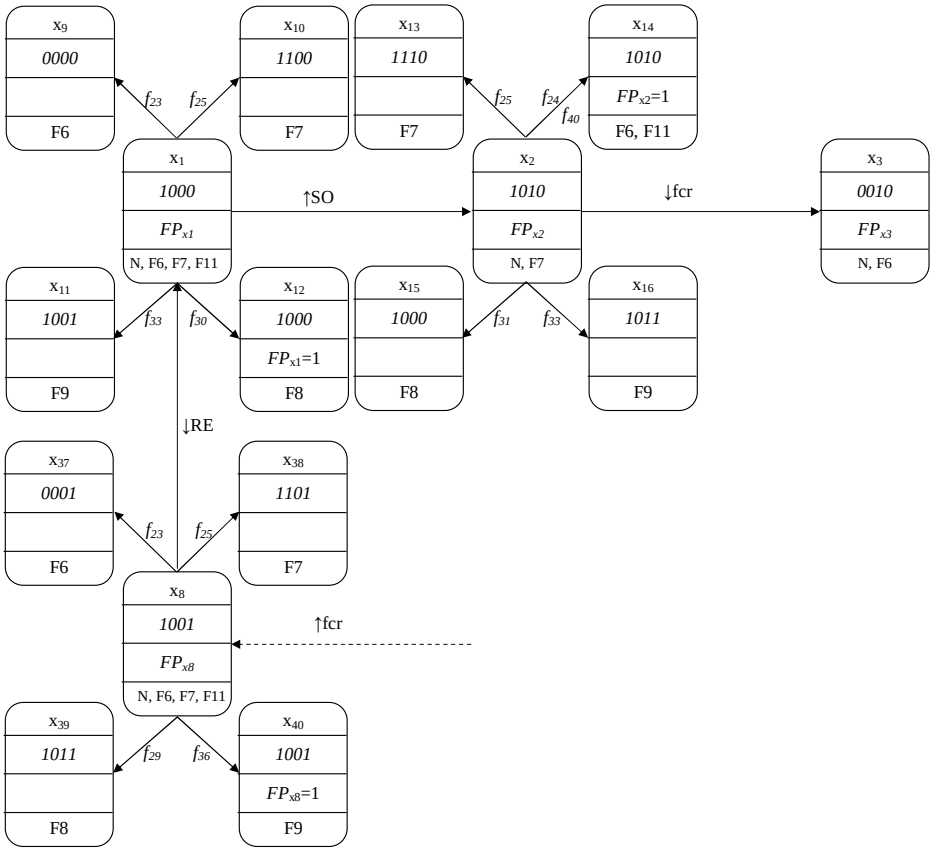


Figure 4-14 : Extrait du Diagnostiqueur local D_1 du vérin

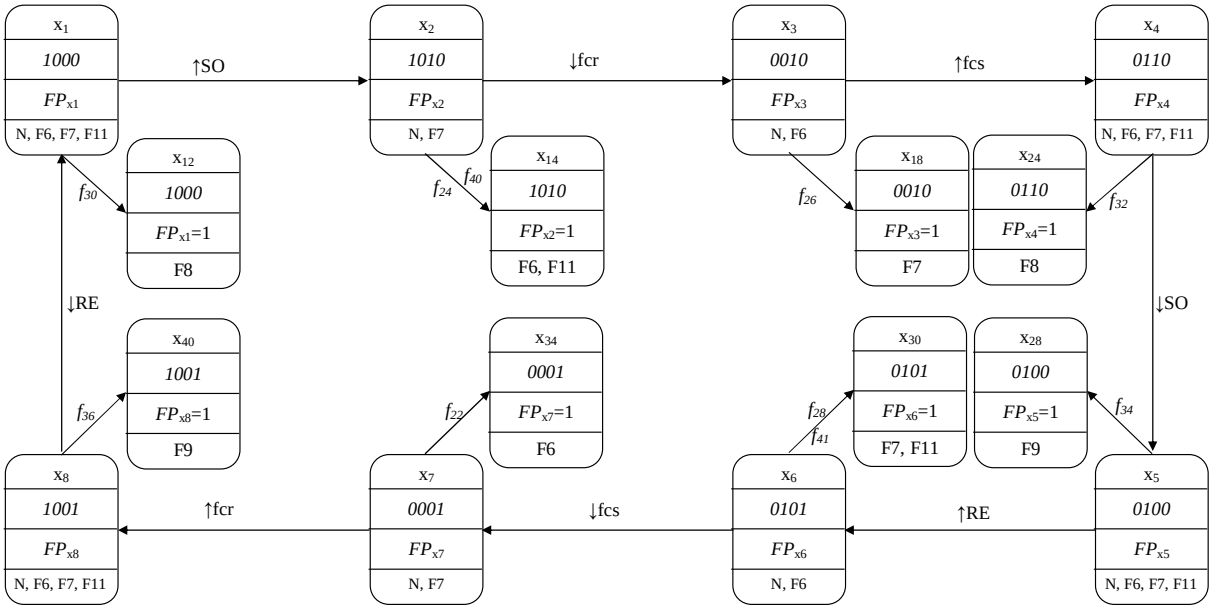


Figure 4-15 : Diagnostiqueur local D_1 du vérin simplifié aux défauts non observables

En ce qui concerne le diagnostiqueur local du chariot D_2, il est possible de détecter un défaut des 6 partitions de défauts Π_{F1}, Π_{F2}, Π_{F3}, Π_{F4}, Π_{F5} et Π_{F10} à partir de tout état normal obtenu par l'$EPOC_2$. Il en résulte un automate enrichi de diagnostiqueur D_2 à 14 états normaux et $14 \times 5 = 70$ états de défauts, soit un total de 84 états pour le diagnostiqueur D_2. Le diagnostiqueur local du chariot D_2 n'est pas détaillé ici.

4.2.7.4. Règles du coordinateur

Une fois les diagnostiqueurs locaux obtenus, la suite de l'approche consiste à construire un coordinateur pour la structure décentralisée afin d'établir les contraintes globales qui ne peuvent pas être prises en compte par un des diagnostiqueurs locaux et de gérer les éventuelles ambiguïtés et cas d'indécision.

Pour l'exemple du wagonnet de la Figure 4-1, aucun détecteur n'étant en commun entre les EPO, il n'y a donc pas de risque d'indécision. Par conséquent, nous ne représenterons pas ici la table de prise de décision finale puisque chaque diagnostiqueur possède sa propre décision. Lorsque les diagnostiqueurs retournent une ambiguïté de décision "-" équivalent à une décision {N, Fi}, l'état appartient alors au comportement désiré sans pour autant garantir l'absence de ce défaut Fi.

Le coordinateur doit exprimer des contraintes globales sur le système à travers des règles devant être respectées à tout moment. Ces règles ne peuvent pas être vérifiées au niveau local et doivent être observées par le coordinateur. Sur l'exemple du wagonnet de Figure 4-1, quatre contraintes peuvent être exprimées (Tableau 4-2). La règle 1 exprime le fait que l'activation de l'ordre de sortie *SO* du vérin par la commande ne peut se faire que sous la condition que le chariot soit sur le détecteur *b* et qu'aucun ordre de déplacement n'est enclenché afin d'empêcher une cassure mécanique d'un vérin. Si cette condition n'est pas respectée, la règle retourne l'étiquette F8 correspondant à un défaut de la partition Π_{F8} lié à l'ordre de sortie du vérin. La règle 2 décrit les mêmes conditions pour l'activation de l'ordre de rentrée *RE* du vérin provenant de la commande implantée avec une étiquette F9. Les règles 3 et 4 décrivent les conditions sur le vérin lors de l'activation des ordres de déplacement du chariot avec respectivement une étiquette F4 et F5.

	Ordre de la commande implantée en ligne	**Conditions à respecter**	**Etiquette**
Règle 1	↑SO	$b \wedge \neg D \wedge \neg G = 1$	F8
Règle 2	↑RE	$b \wedge \neg D \wedge \neg G = 1$	F9
Règle 3	↑D	$fcr \wedge \neg SO \wedge \neg RE = 1$	F4
Règle 4	↑G	$fcr \wedge \neg SO \wedge \neg RE = 1$	F5

Tableau 4-2 : Règles du coordinateur pour l'exemple du wagonnet

4.2.8. Comparaison des structures

Pour illustrer la comparaison entre les deux structures, nous reprenons l'ensemble des automates dans le Tableau 4-3. Nous nous apercevons tout d'abord que l'explosion combinatoire est évitée au niveau de la structure décentralisée pour la PO. En effet, pour obtenir un diagnostiqueur global, il faut établir un modèle global de la PO par composition synchrone de l'ensemble des EPO alors que pour une structure décentralisée, il suffit d'obtenir les modèles élémentaires de la PO pour obtenir les diagnostiqueurs locaux. Dès lors, le nombre d'états croît de façon exponentielle dans le cas de structure centralisée en fonction du nombre de PO.

Une deuxième remarque de même type peut être faite sur le modèle de PO Commandée et le diagnostiqueur final. La structure centralisée retourne un diagnostiqueur global comportant

220 états alors que le nombre d'états pour le diagnostiqueur local D_1 est de 40, et de 84 pour le diagnostiqueur local D_2.

Enfin, cet exemple permet de mesurer l'impact des deux algorithmes d'intersection. L'algorithme 1 implique un passage de composition synchrone entre deux modèles évolués ce qui a tendance à augmenter le nombre d'états sur l'automate résultant. Le fait de restreindre un automate avant la composition synchrone entraîne, pour l'algorithme 2, une diminution du risque d'explosion combinatoire.

Pour l'exemple du wagonnet de la Figure 4-1, l'utilisation de l'algorithme 1 d'intersection engendre le passage de composition synchrone à 18 états pour le vérin et 21 états pour le chariot. L'algorithme 2 n'excède pas, quant à lui, les 8 états de l'$EPOC_1$ du vérin et 14 états de l'$EPOC_2$ du chariot. En conclusion, nous voyons que pour le même résultat, il y a moins de risque à l'explosion combinatoire en utilisant l'algorithme 2 d'intersection.

	Centralisée	**Décentralisée**			
		Algorithme 1		Algorithme 2	
		Vérin	Chariot	Vérin	Chariot
GRAFCET	6 étapes	6 étapes		6 étapes	
GE	16 états	16 états		16 états	
PO	450 états	15 états	30 états	15 états	30 états
GE || PO	23 états	18 états	21 états	X	X
GE Restreint	15 états	X	X	6 états	9 états
POC	22 états	8 états	14 états	8 états	14 états
Diagnostiqueur	220 états	40 états	84 états	40 états	84 états

Tableau 4-3 : Comparatif entre la structure centralisée et décentralisée pour l'exemple de la Figure 4-1

Nous venons de voir l'intérêt de la structure décentralisée par rapport à la structure centralisée en terme d'états, mais peut-on garantir les mêmes performances en terme de diagnosticabilité ? Pour répondre à cette question, il faut étudier la notion de co-diagnosticabilité mixte que nous avons établie dans le chapitre 3. La structure centralisée permet de diagnostiquer les partitions de défauts Π_{F1}, Π_{F2}, Π_{F3}, Π_{F4}, Π_{F5}, Π_{F6}, Π_{F7}, Π_{F8}, Π_{F9}, Π_{F10} et Π_{F11}. Il faut que l'ensemble des diagnostiqueurs locaux de la structure décentralisée puisse diagnostiquer l'ensemble de ces partitions pour que le système soit co-diagnosticable. Le diagnostiqueur D_1 permet de diagnostiquer les partitions Π_{F6}, Π_{F7}, Π_{F8}, Π_{F9} et Π_{F11} tandis que le diagnostiqueur D_2 permet de diagnostiquer les partitions Π_{F1}, Π_{F2}, Π_{F3}, Π_{F4}, Π_{F5} et Π_{F10}. Dès lors, l'ensemble des partitions est bien diagnostiqué par la structure décentralisée.

Les matrices de défauts du vérin, MF_1, et du chariot, MF_2, sont données respectivement par les Tableau 4-4 et Tableau 4-5. Elles permettent ainsi de définir le retard maximum de diagnostic de défauts non observables par $RD_{max} = max(RD_{1max}, RD_{2max}) = max(4, 10) = 10$. On peut s'apercevoir également que ces matrices permettent de diagnostiquer l'ensemble des partitions de défauts et qu'elles ne possèdent aucune boucle d'incertitude. Par conséquent, le

système est considéré comme co-diagnosticable avec un retard maximum de détection de 10 événements observables. De plus, chaque colonne de la matrice rappelle par ces "0" les défauts qu'il n'est pas possible de diagnostiquer pour chaque état.

	Etat / Défaut	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
F6	f_{22}	1	1	0	0	0	0	1	1
	f_{24}	0	1	1	1	1	1	1	0
F7	f_{26}	0	0	1	1	1	1	0	0
	f_{28}	1	1	1	0	0	1	1	1
F8	f_{30}	1	1	1	1	1	1	1	1
	f_{32}	1	1	1	1	1	1	1	1
F9	f_{34}	1	1	1	1	1	1	1	1
	f_{36}	1	1	1	1	1	1	1	1
F11	f_{40}	0	1	1	1	1	1	1	0
	f_{41}	1	1	1	0	0	1	1	1

Tableau 4-4 : Matrice de défauts, MF_1, du vérin

	Etat / Défaut	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
F1	f_2	1	1	0	0	0	0	0	0	0	0	0	0	1	1
	f_4	0	1	1	1	1	1	1	1	1	1	1	1	1	0
F2	f_6	0	0	1	1	0	0	0	0	1	1	1	1	0	0
	f_8	1	1	1	0	1	1	1	1	1	0	0	0	1	1
F3	f_{10}	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	f_{12}	1	1	1	1	1	0	0	1	1	1	1	1	1	1
F4	f_{14}	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	f_{16}	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F5	f_{18}	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	f_{20}	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F10	f_{37}	0	1	1	1	1	1	1	1	1	1	1	1	1	0
	f_{38}	1	1	1	1	1	1	1	1	1	0	0	1	1	1
	f_{39}	1	1	1	1	1	0	0	1	1	1	1	1	1	1

Tableau 4-5 : Matrice de défauts, MF_2, du chariot

Les performances de co-diagnosticabilité de la structure décentralisée s'expliquent par la capacité à diagnostiquer localement l'ensemble des défauts diagnosticables par un diagnostiqueur global. De plus, pour cet exemple simple, il n'y a pas de détecteurs communs entre les deux diagnostiqueurs. Par conséquent, il n'y a aucune partition qu'un diagnostiqueur local ne puisse diagnostiquer. La prise de décision est donc équivalente entre la structure centralisée et décentralisée avec coordinateur exprimant les contraintes globales.

4.3. Exemple 2 : Système de tri de caisses

Nous venons de voir sur un petit exemple simple, la justification de la structure décentralisée choisie pour notre diagnostic. Cependant, quelles sont les réelles difficultés sur

une application manufacturière de plus grande taille ? Pour cela, nous allons étudier un système manufacturier composé de plusieurs actionneurs et répondant à des contraintes de fonctionnement particulières, notamment par l'utilisation de capteurs communs à plusieurs actionneurs.

4.3.1. Présentation de l'exemple

4.3.1.1. Technologie

Un dispositif automatique destiné à trier deux types de caisses est présenté en Figure 4-16. Ce système est composé d'un tapis amenant des caisses, de trois vérins poussoirs et de deux tapis d'évacuation.

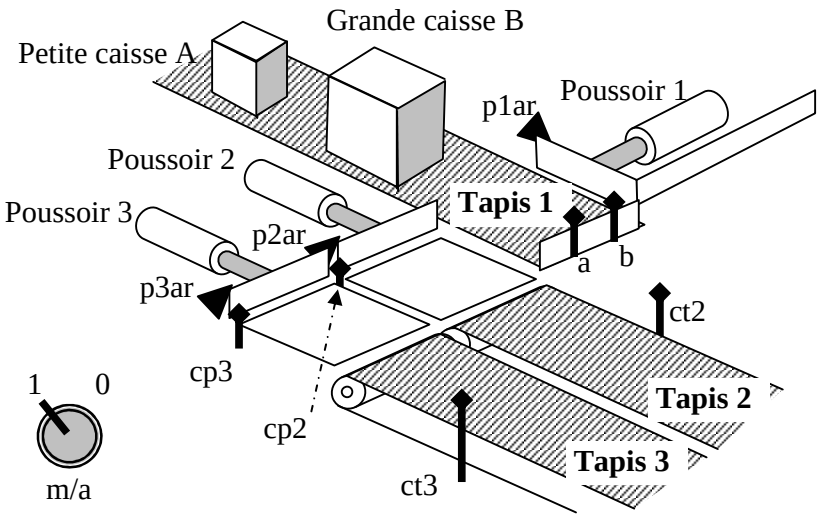


Figure 4-16 : Système de tri de caisses

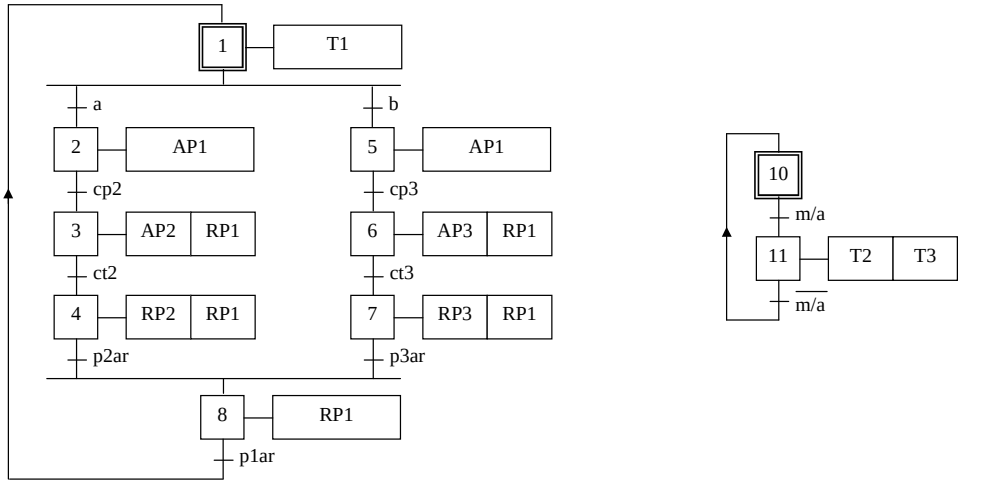
Ce système est composé de 6 EPO aux technologies différentes :

- Le poussoir 1 est un VDE piloté par un distributeur 5/3 Centre Fermé. Il est entouré de trois détecteurs : *p1ar* est le fin de course rentrée, *cp2* détecte la position intermédiaire et *cp3* est le fin de course de sortie. La sortie de la tige du vérin est commandée par l'ordre AP1 et sa rentrée par l'ordre RP1.
- Les poussoirs 2 et 3 sont des VDE pilotés par des distributeurs 5/2 bistables. Ils possèdent un détecteur de fin de course rentrée (*p2ar* et *p3ar*) et leur sortie est conditionnée par le passage de la caisse sur le détecteur du tapis (*ct2* et *ct3*). Chaque poussoir *i* possède un ordre de sortie de tige du poussoir APi et un ordre de rentrée de tige RPi.
- Le tapis 1 est un moteur à un sens de rotation commandé par l'ordre T1. Ce tapis est en rotation tant qu'aucune caisse n'est détectée par le détecteur *a* ou *b*. La présence d'une caisse sur l'un de ces détecteurs implique l'arrêt du moteur. Il est à noter que la sortie du poussoir 1 entraîne, pour des raisons de sécurité évidentes, l'arrêt du moteur. Dès lors, le tapis n'est en rotation que si le poussoir 1 est en *p1ar* et si aucune pièce n'est en *a* ou en *b*.
- Les tapis 2 et 3 sont des moteurs à un sens de rotation commandés par un ordre T2 pour le tapis 2 et T3 pour le tapis 3. Les moteurs de ces tapis sont en rotation

permanente. Seul l'arrêt complet du système engendre l'arrêt de ces tapis. Un détecteur de présence de caisses (*ct2* et *ct3*) est positionné sur chacun des tapis. Ces détecteurs se trouvent activés à chaque passage de caisse. On remarque donc que le détecteur *ct2*, respectivement *ct3*, est utilisé à la fois pour le poussoir 2, respectivement 3, et pour le tapis 2, respectivement 3.

4.3.1.2. Cahier des charges

Le tapis 1 amène les caisses devant le poussoir 1. Pour effectuer la sélection des caisses, un dispositif de détection a été placé au bout du tapis 1 permettant de reconnaître une petite caisse par le détecteur $a = 1$ et une grande caisse par le détecteur $b = 1$. Le poussoir 1 pousse les petites caisses jusqu'au détecteur *cp2* pour les mettre devant le poussoir 2 qui les transfère ensuite sur le tapis 2. Le détecteur *ct2* confirme alors le passage d'une caisse sur le tapis 2. Les grandes caisses sont poussées jusqu'au détecteur *cp3* devant le poussoir 3 qui les évacuent sur le tapis 3 où le détecteur *ct3* confirme leur passage. La modélisation de la commande est présentée Figure 4-17.



a) GRAFCET du système de tri

b) GRAFCET des tapis 2 et 3 du système de tri

Figure 4-17 : Commande du système de tri de caisses

La commande des tapis 2 et 3 est représentée par un GRAFCET à deux étapes (Figure 4-17b) où à partir de l'étape initiale (étape 10), il faut attendre la mise en marche par le sélecteur (m/a = 1) pour activer la rotation des tapis par les ordres T2 et T3 (étape 11). L'arrêt du système entraîne l'arrêt de la rotation et donc le retour en étape initiale. La commande de tri proprement dite est représentée par le GRAFCET de la Figure 4-17a. Dans l'étape initiale (étape 1), le tapis 1 est mis en rotation par l'ordre T1. Dès lors, il faut attendre soit la présence d'une petite caisse qui implique un passage à l'étape 2, soit la présence d'une grande caisse ce qui va activer l'étape 5.

Une petite caisse implique l'arrêt du tapis 1 et l'activation de la sortie du poussoir 2 (étape 2) jusqu'au détecteur *cp2*. Dans ce cas, le poussoir 2 doit éjecter la caisse sur le tapis 2 par l'ordre AP2 pendant que le poussoir 1 retourne en position de rentrée par l'ordre RP1 (étape 3). Lorsque la petite caisse passe devant le capteur *ct2*, alors il faut rentrer le poussoir 2 jusqu'à la position *p2ar* (étape 4). L'étape 8 attend la confirmation que les vérins sont tous en position rentrée pour réactiver le tapis 1.

La détection d'une grande caisse engendre un fonctionnement quasi similaire. Une grande

caisse provoque l'arrêt du tapis 1 et l'activation de la sortie du poussoir 3 (étape 5) jusqu'au détecteur *cp3*. L'ordre de sortie AP3 du poussoir 3 éjecte la caisse sur le tapis 3 alors que le poussoir 1 commence son retour par l'ordre RP1 (étape 6). Lorsque la grande caisse passe devant le capteur *ct3*, alors il faut rentrer le poussoir 3 jusqu'à sa position *p3ar* (étape 7).

4.3.2. Etablissement des modèles EPO

4.3.2.1. Poussoir 1

Après avoir identifié chaque EPO, il faut obtenir le modèle correspondant. Le vérin poussoir 1 est un VDE double effet piloté par un distributeur 5/3 Centre Fermé entouré de trois détecteurs ce qui donne un modèle des détecteurs à 6 états (Figure 4-18a) et un modèle actionneur à 5 états ne permettant pas le déplacement de la tige du vérin en position intermédiaire (Figure 4-18b) établi dans la bibliothèque de l'annexe 2. Il en résulte un automate à 30 états représentant l'EPO_{P1} du poussoir 1 (Figure 4-18c).

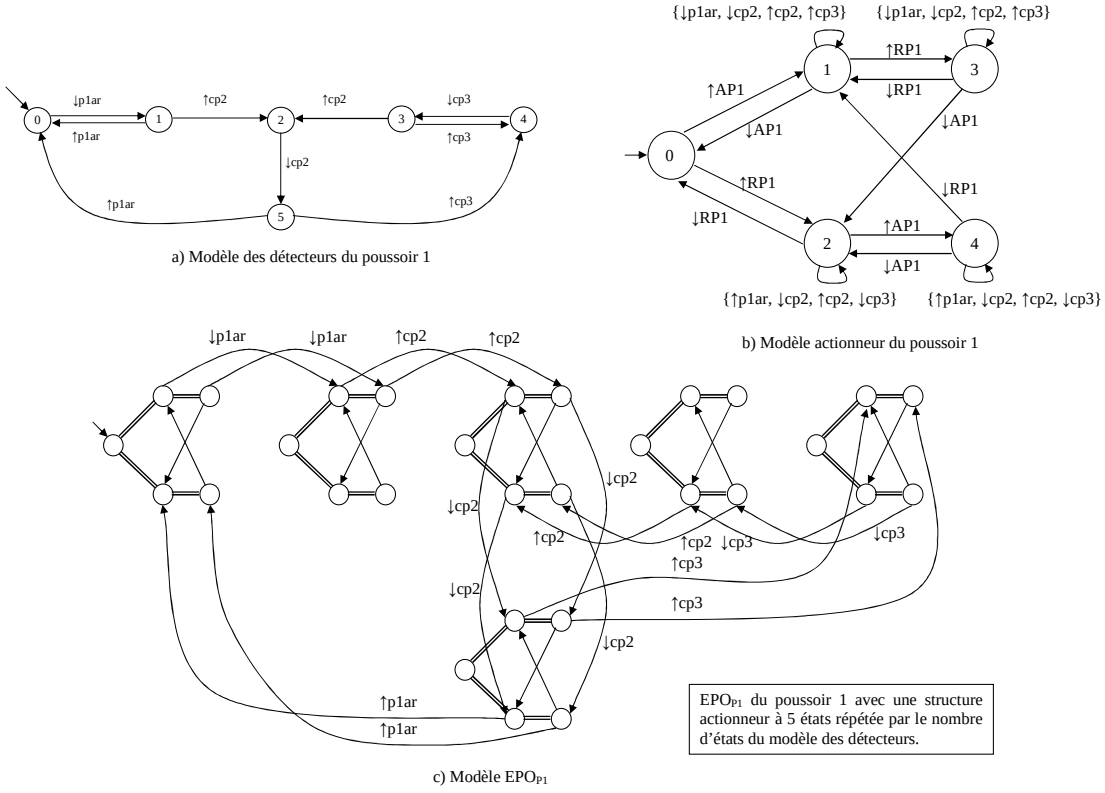


Figure 4-18 : Construction de l'Elément de Partie Opérative EPO_{P1} du poussoir 1

4.3.2.2. Poussoirs 2 et 3

Le vérin poussoir 2 est un VDE double effet piloté par un distributeur 5/2 bistable entouré de deux détecteurs. La construction du modèle de l'EPO_{P2} du poussoir 2 est représentée en Figure 4-19. Le vérin réagit à un détecteur de fin de course *p2ar* mais aussi au détecteur de présence de caisse *ct2*. Dès lors, son modèle va comporter quelques particularités sur son modèle des détecteurs et son modèle actionneur. En effet, le détecteur *ct2* est enclenché par une caisse et non pas par la sortie du vérin en fin de course. Ceci implique également que la désactivation de ce détecteur dépend de la caisse, donc du produit, et non pas de la tige du

vérin. Dès lors, même en position de sortie complète du vérin, le détecteur *ct2* peut se désactiver. De même, pendant la rentrée de la tige, il est possible de rencontrer la désactivation de *ct2*. Ces particularités s'expriment sur le modèle des détecteurs, Figure 4-19a, par un automate à 4 états. A l'état 0, le vérin est rentré et il faut attendre sa sortie pour désactiver *p2ar* (état 1). Une caisse sur le détecteur *ct2* engendre son activation (état 2). A partir de cet état, soit le détecteur se désactive avant la rentrée complète du vérin (retour en état 1 puis 0), soit le poussoir est revenu en position de rentrée avant la désactivation du détecteur *ct2* (état 3 puis 0). Le modèle actionneur se trouve également changé au niveau de ces boucles d'autorisation d'événements non commandables (Figure 4-19b). Ainsi, dans les états 1 et 3 de cet automate, il faut autoriser la désactivation de *p1ar*, l'activation de *ct2* mais également la désactivation de *ct2*. La composition synchrone de l'automate des détecteurs avec celui de l'actionneur fournit le modèle EPO_{P2} de la Figure 4-19c. Ce modèle est donc un nouvel élément de notre bibliothèque d'EPO.

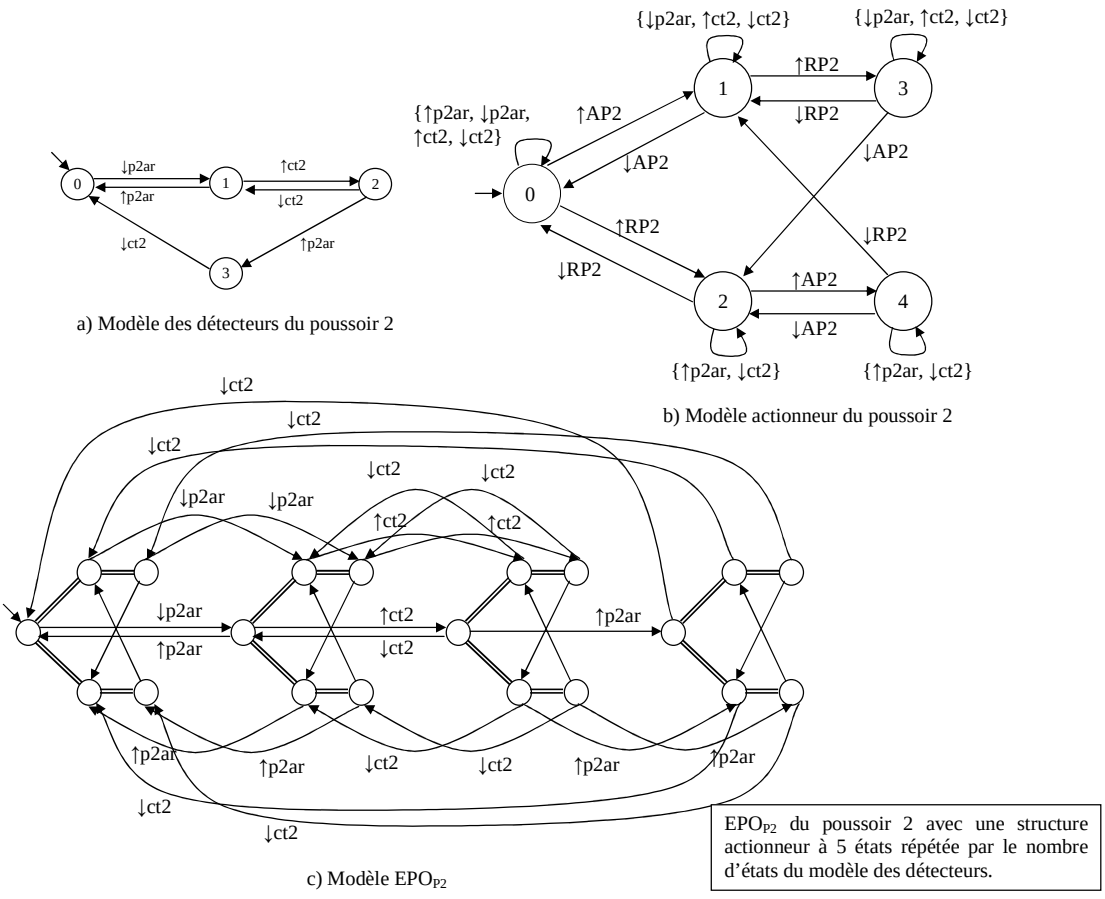


Figure 4-19 : Construction de l'Elément de Partie Opérative EPO_{P2} du poussoir 2

Le vérin poussoir 3 étant de même nature que le vérin poussoir 2, le modèle de l'EPO_{P3} final est donc identique (Figure 4-20).

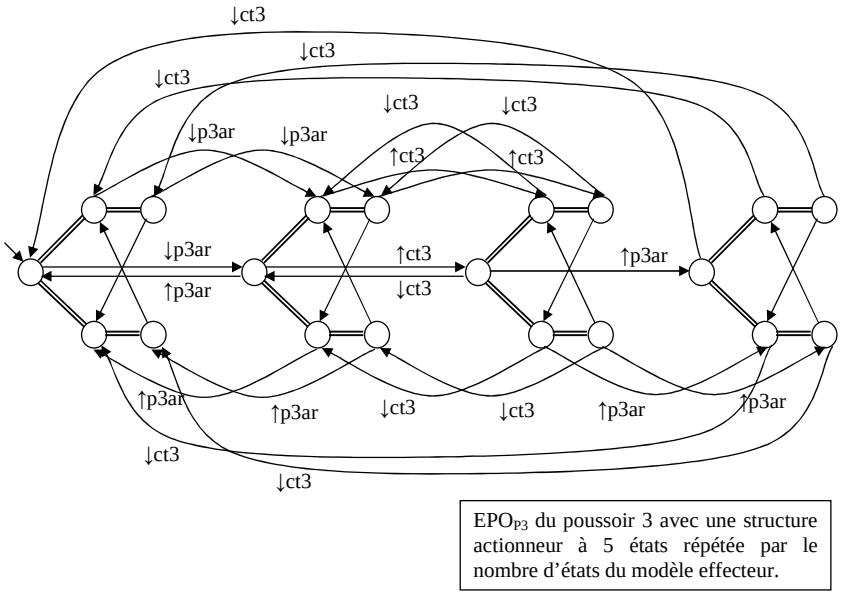
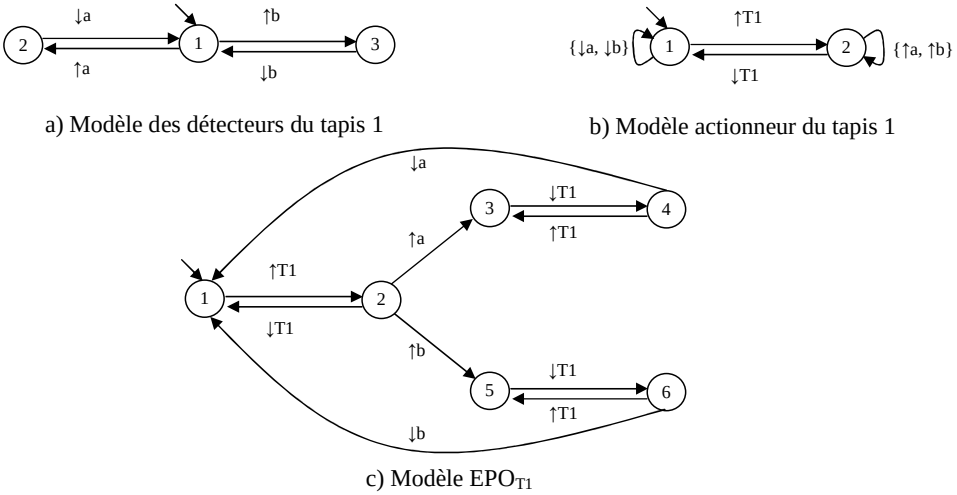


Figure 4-20 : Elément de Partie Opérative EPO_{P3} du poussoir 3

4.3.2.3. Tapis 1

Le tapis 1 est un moteur à un sens de rotation et doit s'arrêter lorsqu'une caisse est présente sur le détecteur *a* ou *b*. Chaque détecteur est affecté à une caisse et ne peut être enclenché simultanément. Il en résulte donc le modèle des détecteurs de la Figure 4-21a à trois états. Le modèle actionneur est un automate à deux états (Figure 4-21b) où lorsque le tapis est à l'arrêt, il est possible de désactiver les détecteurs *a* et *b* (état 1). Par contre, lorsque le tapis est en rotation, il est possible d'avoir l'activation de *a* ou *b* (état 2). Au final, le modèle d'EPO_{T1} du tapis 1 est un automate à 6 états représenté Figure 4-21c où de l'état 1, il faut attendre l'activation du tapis (état 2) pour attendre soit une petite caisse par l'activation de *a* (état 3), soit une grosse caisse par l'activation de *b* (état 5). Dans les deux cas, il faut ensuite désactiver la rotation du tapis (états 4 et 6) pour avoir la désactivation des détecteurs et revenir ainsi en situation initiale.



a) Modèle des détecteurs du tapis 1

b) Modèle actionneur du tapis 1

c) Modèle EPO_{T1}

Figure 4-21 : Construction de l'Elément de Partie Opérative EPO_{T1} du tapis 1

4.3.2.4. Tapis 2 et 3

Le tapis 2 et 3 sont identiques et réagissent de la même manière. Il n'y a qu'un seul détecteur indiquant la présence d'une caisse sur le tapis ce qui implique un modèle des détecteurs à deux états (Figure 4-22a et d). Etant des tapis à un seul sens de rotation, ils sont commandés par un seul ordre en activation et en désactivation. Par conséquent, le modèle actionneur est un automate à deux états où les événements non commandables ne peuvent survenir quand il y a rotation (Figure 4-22b et e). La composition synchrone de l'automate des détecteurs avec celui de l'actionneur donne un automate à 4 états représentant l'EPO_{T2} et l'EPO_{T3} des deux tapis (Figure 4-22c et f). De l'état 1 où le moteur n'est pas en fonctionnement, il est possible d'aller à l'état 2 par l'activation de l'ordre de rotation. A partir de cet état, soit l'ordre est désactivé, soit une caisse est détectée (état 3). Dès lors, il est possible d'arrêter le tapis (état 4) ou d'attendre la désactivation du détecteur (retour en état 2).

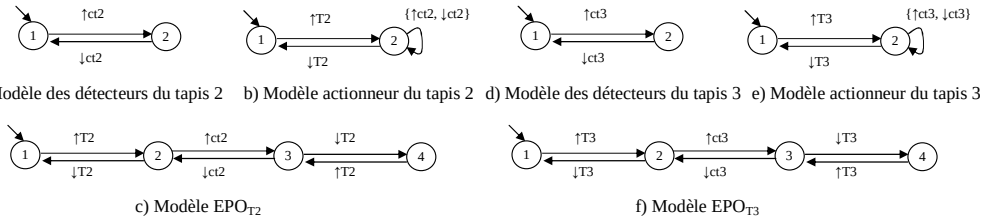


Figure 4-22 : Construction des Eléments de Partie Opérative EPO_{T2} et EPO_{T3} des tapis 2 et 3

C'est à partir de ces différents EPO que l'information provenant de la commande est intégrée permettant d'obtenir le comportement désiré de chacun des EPO. Auparavant, à partir de chaque EPO_i, il faut établir l'ensemble des partitions de défauts qu'il est possible de diagnostiquer.

4.3.3. Partitions de défauts

Pour chaque détecteur présent sur l'EPO_i, l'ensemble des défauts possibles est présenté dans le Tableau 4-6.

Défaut détecteur observable	Passage inattendu d'un détecteur de 0 à 1
	Passage inattendu d'un détecteur de 1 à 0
Défaut détecteur non observable	Sortie d'un détecteur bloqué au niveau 0
	Sortie d'un détecteur bloqué au niveau 1
Erreur sur un ordre envoyé par la PC	Passage inattendu d'un ordre de 0 à 1
	Passage inattendu d'un ordre de 1 à 0
Erreur sur un ordre non envoyé par la PC	Attende d'activation d'un non envoyée par la PC
	Attente de désactivation d'un non envoyée par la PC
Défaut actionneur non observable	Actionneur bloqué inactif
	Actionneur bloqué actif

Tableau 4-6 : Types de défauts possibles sur un EPO_i

Les défauts observables sont détectés trivialement sans aucun retard de détection. Les défauts non observables sont, quant à eux, diagnostiqués avec un retard de détection par les fonctions de prévisions que nous présentons dans le paragraphe 4.3.7.

4.3.3.1. Partitions de défauts liées au poussoir 1

Le diagnostiqueur D_{P1} doit diagnostiquer les différents défauts référencés en Tableau 4-7. Il est possible de les classifier par les partitions $\Sigma_{\Pi DP1} = \{\Pi_{F1}, \Pi_{F2}, \Pi_{F3}, \Pi_{F4}, \Pi_{F5}, \Pi_{F6}\}$:

- Défauts sur le détecteur *p1ar* : $\Pi_{F1} = \{f_1, f_2, f_3, f_4\}$ avec l'étiquette F1,
- Défauts sur le détecteur *cp2* : $\Pi_{F2} = \{f_5, f_6, f_7, f_8\}$ avec l'étiquette F2,
- Défauts sur le détecteur *cp3* : $\Pi_{F3} = \{f_9, f_{10}, f_{11}, f_{12}\}$ avec l'étiquette F3,
- Défauts de la PC liés à l'ordre de sortie AP1 : $\Pi_{F4} = \{f_{13}, f_{14}, f_{15}, f_{16}\}$ avec l'étiquette F4,
- Défauts de la PC liés à l'ordre de rentrée RP1 : $\Pi_{F5} = \{f_{17}, f_{18}, f_{19}, f_{20}\}$ avec l'étiquette F5,
- Défauts sur le poussoir P1 bloqué : $\Pi_{F6} = \{f_{21}, f_{22}\}$ avec l'étiquette F6.

F1	f_1	Passage inattendu de *p1ar* de 0 à 1	F4	f_{13}	Passage inattendu de AP1 de 0 à 1
	f_2	Détecteur *p1ar* bloqué à 0		f_{14}	Activation de AP1 non envoyée
	f_3	Passage inattendu de *p1ar* de 1 à 0		f_{15}	Passage inattendu de AP1 de 1 à 0
	f_4	Détecteur *p1ar* bloqué à 1		f_{16}	Désactivation de AP1 non envoyée
F2	f_5	Passage inattendu de *cp2* de 0 à 1	F5	f_{17}	Passage inattendu de RP1 de 0 à 1
	f_6	Détecteur *cp2* bloqué à 0		f_{18}	Activation de RP1 non envoyée
	f_7	Passage inattendu de *cp2* de 1 à 0		f_{19}	Passage inattendu de RP1 de 1 à 0
	f_8	Détecteur *cp2* bloqué à 1		f_{20}	Désactivation de RP1 non envoyée
F3	f_9	Passage inattendu de *cp3* de 0 à 1	F6	f_{21}	Actionneur bloqué en position rentrée
	f_{10}	Détecteur *cp3* bloqué à 0		f_{22}	Actionneur bloqué en position sortie
	f_{11}	Passage inattendu de *cp3* de 1 à 0			
	f_{12}	Détecteur *cp3* bloqué à 1			

Tableau 4-7 : Défauts retrouvés sur le vérin poussoir 1

4.3.3.2. Partitions de défauts liées au poussoir 2

Pour le diagnostiqueur du poussoir 2, D_{P2}, il est possible de classifier les défauts que l'on peut diagnostiquer (Tableau 4-8) en fonction de leur nature $\Sigma_{\Pi DP2} = \{\Pi_{F7}, \Pi_{F8}, \Pi_{F9}, \Pi_{F10}, \Pi_{F11}\}$:

- Défauts sur le détecteur *p2ar* : $\Pi_{F7} = \{f_{23}, f_{24}, f_{25}, f_{26}\}$ avec l'étiquette F7,
- Défauts sur le détecteur *ct2* : $\Pi_{F8} = \{f_{27}, f_{28}, f_{29}, f_{30}\}$ avec l'étiquette F8,
- Défauts de la PC liés à l'ordre de sortie AP2 : $\Pi_{F9} = \{f_{31}, f_{32}, f_{33}, f_{34}\}$ avec l'étiquette F9,
- Défauts de la PC liés à l'ordre de rentrée RP2 : $\Pi_{F10} = \{f_{35}, f_{36}, f_{37}, f_{38}\}$ avec

l'étiquette F10,

- Défauts sur le poussoir P2 bloqué : $\Pi_{F11} = \{f_{39}, f_{40}\}$ avec l'étiquette F11.

F7	f_{23}	Passage inattendu de *p2ar* de 0 à 1	F9	f_{31}	Passage inattendu de AP2 de 0 à 1
	f_{24}	Détecteur *p2ar* bloqué à 0		f_{32}	Activation de AP2 non envoyée
	f_{25}	Passage inattendu de *p2ar* de 1 à 0		f_{33}	Passage inattendu de AP2 de 1 à 0
	f_{26}	Détecteur *p2ar* bloqué à 1		f_{34}	Désactivation de AP2 non envoyée
F8	f_{27}	Passage inattendu de *ct2* de 0 à 1	F10	f_{35}	Passage inattendu de RP2 de 0 à 1
	f_{28}	Détecteur *ct2* bloqué à 0		f_{36}	Activation de RP2 non envoyée
	f_{29}	Passage inattendu de *ct2* de 1 à 0		f_{37}	Passage inattendu de RP2 de 1 à 0
	f_{30}	Détecteur *ct2* bloqué à 1		f_{38}	Désactivation de RP2 non envoyée
			F11	f_{39}	Actionneur bloqué en position rentrée
				f_{40}	Actionneur bloqué en position sortie

Tableau 4-8 : Défauts retrouvés sur le vérin poussoir 2

4.3.3.3. Partitions de défauts liées au poussoir 3

Le diagnostiqueur du poussoir 3, D_{P3}, permet de diagnostiquer les partitions de défauts $\Sigma_{\Pi DP3} = \{\Pi_{F12}, \Pi_{F13}, \Pi_{F14}, \Pi_{F15}, \Pi_{F16}\}$ (Tableau 4-9) :

- Défauts sur le détecteur *p3ar* : $\Pi_{F12} = \{f_{41}, f_{42}, f_{43}, f_{44}\}$ avec l'étiquette F12,
- Défauts sur le détecteur *ct3* : $\Pi_{F13} = \{f_{45}, f_{46}, f_{47}, f_{48}\}$ avec l'étiquette F13,
- Défauts de la PC liés à l'ordre de sortie AP3 : $\Pi_{F14} = \{f_{49}, f_{50}, f_{51}, f_{52}\}$ avec l'étiquette F14,
- Défauts de la PC liés à l'ordre de rentrée RP3 : $\Pi_{F15} = \{f_{53}, f_{54}, f_{55}, f_{56}\}$ avec l'étiquette F15,
- Défauts sur le poussoir P3 bloqué : $\Pi_{F16} = \{f_{57}, f_{58}\}$ avec l'étiquette F16.

F12	f_{41}	Passage inattendu de *p3ar* de 0 à 1	F14	f_{49}	Passage inattendu de AP3 de 0 à 1
	f_{42}	Détecteur *p3ar* bloqué à 0		f_{50}	Activation de AP3 non envoyée
	f_{43}	Passage inattendu de *p3ar* de 1 à 0		f_{51}	Passage inattendu de AP3 de 1 à 0
	f_{44}	Détecteur *p3ar* bloqué à 1		f_{52}	Désactivation de AP3 non envoyée
F13	f_{45}	Passage inattendu de *ct3* de 0 à 1	F15	f_{53}	Passage inattendu de RP3 de 0 à 1
	f_{46}	Détecteur *ct3* bloqué à 0		f_{54}	Activation de RP3 non envoyée
	f_{47}	Passage inattendu de *ct3* de 1 à 0		f_{55}	Passage inattendu de RP3 de 1 à 0
	f_{48}	Détecteur *ct3* bloqué à 1		f_{56}	Désactivation de RP3 non envoyée
			F16	f_{57}	Actionneur bloqué en position rentrée
				f_{58}	Actionneur bloqué en position sortie

Tableau 4-9 : Défauts retrouvés sur le vérin poussoir 3

4.3.3.4. Partitions de défauts liées au tapis 1

En ce qui concerne le tapis 1, le diagnostiqueur D_{T1} permet de diagnostiquer les partitions de défauts $\Sigma_{\Pi DT1} = \{\Pi_{F17}, \Pi_{F18}, \Pi_{F19}, \Pi_{F20}\}$ (Tableau 4-10) :

- Défauts sur le détecteur *a* : $\Pi_{F17} = \{f_{59}, f_{60}, f_{61}, f_{62}\}$ avec l'étiquette F17,
- Défauts sur le détecteur *b* : $\Pi_{F18} = \{f_{63}, f_{64}, f_{65}, f_{66}\}$ avec l'étiquette F18,
- Défauts de la PC liés à l'ordre de rotation T1 : $\Pi_{F19} = \{f_{67}, f_{68}, f_{69}, f_{70}\}$ avec l'étiquette F19,
- Défauts sur le tapis T1 bloqué à l'arrêt : $\Pi_{F20} = \{f_{71}\}$ avec l'étiquette F20.

F17	f_{59}	Passage inattendu de *a* de 0 à 1	F19	f_{67}	Passage inattendu de T1 de 0 à 1
	f_{60}	Détecteur *a* bloqué à 0		f_{68}	Activation de T1 non envoyée
	f_{61}	Passage inattendu de *a* de 1 à 0		f_{69}	Passage inattendu de T1 de 1 à 0
	f_{62}	Détecteur *a* bloqué à 1		f_{70}	Désactivation de T1 non envoyée
F18	f_{63}	Passage inattendu de *b* de 0 à 1	F20	f_{71}	Actionneur bloqué à l'arrêt
	f_{64}	Détecteur *b* bloqué à 0			
	f_{65}	Passage inattendu de *b* de 1 à 0			
	f_{66}	Détecteur *b* bloqué à 1			

Tableau 4-10 : Défauts retrouvés sur le tapis 1

4.3.3.5. Partitions de défauts liées au tapis 2

Le diagnostiqueur du tapis 2, D_{T2}, a la particularité d'avoir un détecteur commun avec le

poussoir 2. Par conséquent, une partition de défauts sur le détecteur *ct2* est commune à celle définie auparavant. La classification de l'ensemble des défauts (Tableau 4-11) est composée de deux partitions $\Sigma_{\Pi DT2} = \{\Pi_{F8}, \Pi_{F21}, \Pi_{F22}\}$:

- Défauts sur le détecteur *ct2* : $\Pi_{F8} = \{f_{27}, f_{28}, f_{29}, f_{30}\}$ avec l'étiquette F8,
- Défauts de la PC liés à l'ordre de rotation T2 : $\Pi_{F21} = \{f_{72}, f_{73}, f_{74}, f_{75}\}$ avec l'étiquette F21,
- Défauts sur le tapis T2 bloqué à l'arrêt : $\Pi_{F22} = \{f_{76}\}$ avec l'étiquette F22.

F8	f_{27}	Passage inattendu de *ct2* de 0 à 1	F21	f_{72}	Passage inattendu de T2 de 0 à 1
	f_{28}	Détecteur *ct2* bloqué à 0		f_{73}	Activation de T2 non envoyée
	f_{29}	Passage inattendu de *ct2* de 1 à 0		f_{74}	Passage inattendu de T2 de 1 à 0
	f_{30}	Détecteur *ct2* bloqué à 1		f_{75}	Désactivation de T2 non envoyée
			F22	f_{76}	Actionneur bloqué à l'arrêt

Tableau 4-11 : Défauts retrouvés sur le tapis 2

4.3.3.6. Partitions de défauts liées au tapis 3

Le diagnostiqueur du tapis 3, D_{T3}, a la même particularité que celui du tapis 2 puisqu'il possède également un détecteur commun avec le poussoir 3. Une partition de défauts sur le détecteur *ct3* est commune à celle définie auparavant. La classification de l'ensemble des défauts (Tableau 4-12) est composée de deux partitions $\Sigma_{\Pi DT3} = \{\Pi_{F13}, \Pi_{F23}, \Pi_{F24}\}$:

- Défauts sur le détecteur *ct3* : $\Pi_{F13} = \{f_{45}, f_{46}, f_{47}, f_{48}\}$ avec l'étiquette F13,
- Défauts de la PC liés à l'ordre de rotation T3 : $\Pi_{F23} = \{f_{77}, f_{78}, f_{79}, f_{80}\}$ avec l'étiquette F23,
- Défauts sur le tapis T3 bloqué à l'arrêt : $\Pi_{F24} = \{f_{81}\}$ avec l'étiquette F24.

F13	f_{45}	Passage inattendu de *ct3* de 0 à 1	F23	f_{77}	Passage inattendu de T3 de 0 à 1
	f_{46}	Détecteur *ct3* bloqué à 0		f_{78}	Activation de T3 non envoyée
	f_{47}	Passage inattendu de *ct3* de 1 à 0		f_{79}	Passage inattendu de T3 de 1 à 0
	f_{48}	Détecteur *ct3* bloqué à 1		f_{80}	Désactivation de T3 non envoyée
			F24	f_{81}	Actionneur bloqué à l'arrêt

Tableau 4-12 : Défauts retrouvés sur le tapis 3

4.3.4. Extraction du Graphe Equivalent du GRAFCET

Avant de pouvoir intégrer la commande dans les EPO, il faut établir les *GE* des deux GRAFCET de la commande de la Figure 4-17. Le *GE* du GRAFCET du système de tri de la Figure 4-17a est représenté en Figure 4-23a et correspond à la description suivante :

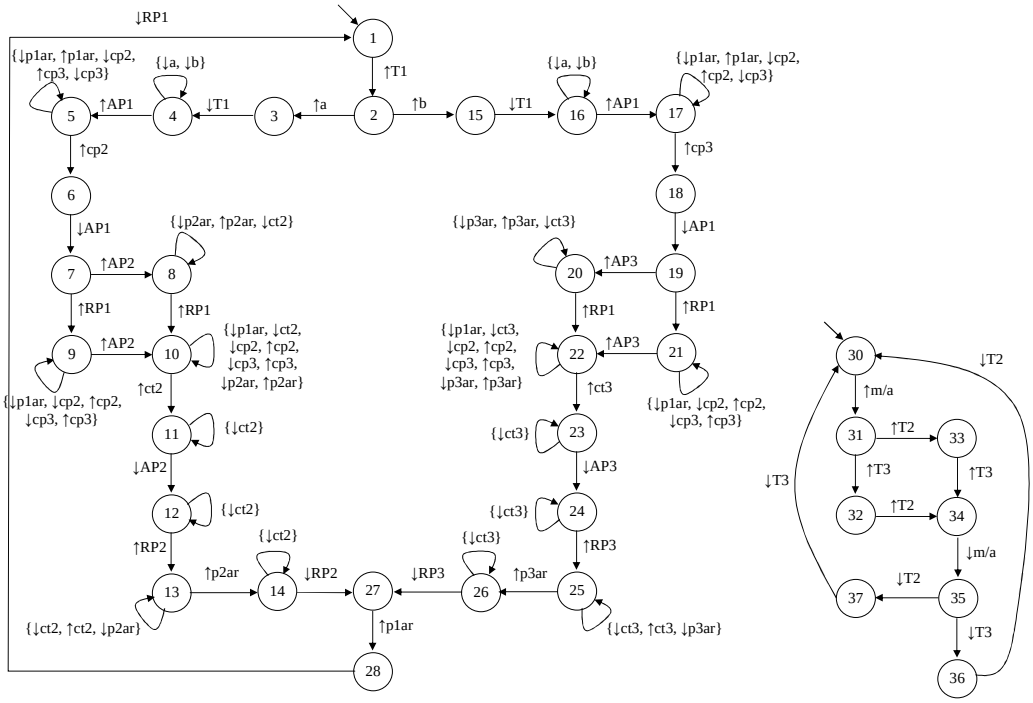
- A partir de l'état 1, il faut envoyer l'ordre T1 correspondant à l'ordre de l'étape 1. De l'état 2, il est possible d'avoir soit l'activation de *a* (état 3), soit l'activation de *b* (état 15) correspondant à la sélection de séquences du GRAFCET.

- De l'état 3, il faut tout d'abord désactiver l'ordre T1 puis activer AP1 (étape 2 du GRAFCET). L'activation de cet ordre engendre l'autorisation d'une boucle d'événements non commandables associés au poussoir 1 sur l'état 5.

- L'activation du détecteur *cp2* permet de passer à l'étape 3 du GRAFCET et par conséquent à l'état 6 du *GE*. Dans cette étape, il faut désactiver AP1 (état 7 du *GE*) puis activer les ordres AP2 et RP1. L'activation de ces deux ordres entraîne la création des états 8, 9 et 10 où, soit l'événement ↑AP2 arrive avant l'événement ↑RP1 (passage vers l'état 8 puis 10), soit l'événement ↑RP1 arrive avant l'événement ↑AP2 (passage vers l'état 9 puis 10). Bien sûr, à chaque activation, correspond un ensemble d'événements non commandables associé à l'ordre autorisé par une boucle sur les états.

- Le détecteur *ct2* permet de quitter l'étape 3 du GRAFCET et donc l'état 10 du *GE* pour se retrouver en état 11. Le procédé se trouve alors en étape 4 où l'ordre RP1 est conservé et l'ordre RP2 est envoyé. Par conséquent, le *GE* commence par la désactivation de l'ordre AP2 (état 12) avant d'activer l'ordre RP2 (état 13). L'ordre RP1 étant conservé sur les étapes 3 et 4 n'implique aucun nouvel état sur le *GE*. L'activation de l'ordre RP2 engendre également une boucle d'événements non commandables sur l'état 13.

- Il faut attendre l'événement ↑p2ar pour quitter l'état 13 vers l'état 14. L'étape 8 du GRAFCET est ensuite activée et implique la désactivation de RP2 (état 27).



a) Graphe Equivalent du GRAFCET du système de tri

b) Graphe Equivalent du GRAFCET des tapis 2 et 3

Figure 4-23 : Graphes Equivalents de la commande

La branche droite du GE représente le comportement du GRAFCET (étape 5, 6 et 7). Pour la grande caisse, il faut attendre *p1ar* pour quitter l'étape 8 du GRAFCET et donc quitter l'état 27 du *GE* pour désactiver l'ordre RP1 (état 28) avant de se retrouver en situation initiale. Il est à noter que dans les états 11, 12, 13 et 15 se trouvent l'événement ↓ct2 dans les boucles permettant la désactivation du détecteur par une caisse. Il en est de même pour les états 23, 24, 25 et 26 avec l'événement ↓ct3. Cette particularité est due au produit et non pas à l'actionneur.

En ce qui concerne les tapis 2 et 3 commandés par le GRAFCET de la Figure 4-17b, le *GE* est donné en Figure 4-23b. Celui-ci décrit un automate à 8 états où, à partir de l'état initial 30, il faut attendre la mise en marche *m/a* pour activer à la fois l'ordre T2 et T3 (état 33 puis 34 ou état 32 puis 34). Les deux ordres restent activés jusqu'à l'arrêt du système par la désactivation de *m/a* où il faut alors désactiver les deux ordres pour se retrouver en situation initiale (état 37 puis 30 ou état 36 puis 30).

4.3.5. Restriction du GE au langage désiré

4.3.5.1. Poussoir 1

L'algorithme 2 d'intersection consiste à effectuer la restriction du *GE* au langage désiré pour l'EPO dont on souhaite obtenir le comportement. Pour le poussoir 1, l'ensemble des événements compris dans son langage est {↓p1ar, ↑p1ar, ↓cp2, ↑cp2, ↓cp3, ↑cp3, ↓AP1, ↑AP1, ↓RP1, ↑RP1}. Dès lors, la restriction du langage sur l'automate *GE* de la Figure 4-23a fournit un automate $GEREPO_{P1}$ à 13 états (Figure 4-24). Chaque état du $GEREPO_{P1}$ correspond à un groupe d'états du *GE* de la Figure 4-23a tels qu'ils sont représentés dans le Tableau 4-13 :

Groupe 1 = {1, 2, 3, 4, 15, 16}	Groupe 8 = {28}
Groupe 2 = {5, 17}	Groupe 9 = {5, 18}
Groupe 3 = {6, 17}	Groupe 10 = {5}
Groupe 4 = {17}	Groupe 11 = {6}
Groupe 5 = {18}	Groupe 12 = {7, 8}
Groupe 6 = {19, 20}	Groupe 13 = {9, 10, 11, 12, 13, 14, 27}
Groupe 7 = {21, 22, 23, 24, 25, 26, 27}	

Tableau 4-13 : Regroupement pour la constitution du $GEREPO_{P1}$

Cet automate décrit le fonctionnement désiré du poussoir 1 qui sera extrait par la composition synchrone grâce au modèle EPO_{P1}.

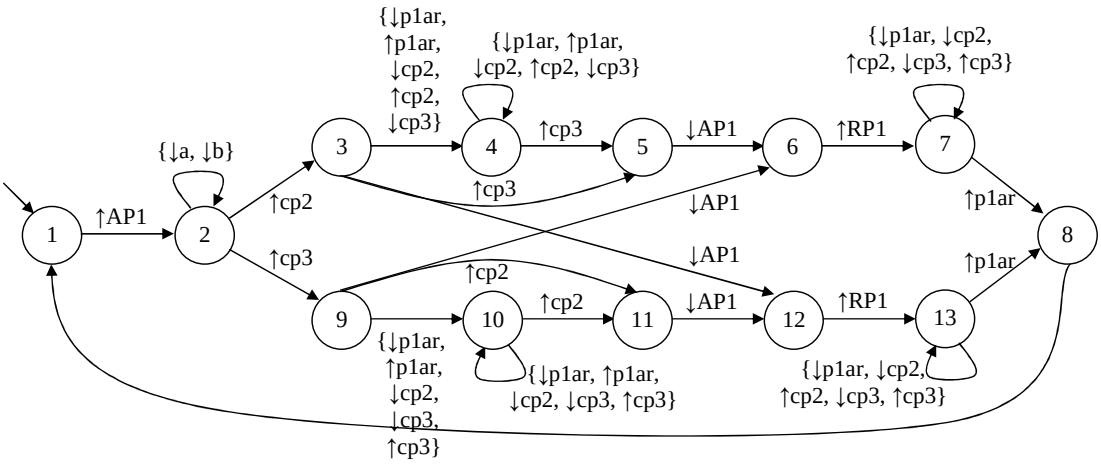


Figure 4-24 : Graphe Equivalent Restreint au langage de l'Elément de Partie Opérative du poussoir 1 : $GEREPO_{P1}$

4.3.5.2. Poussoirs 2 et 3

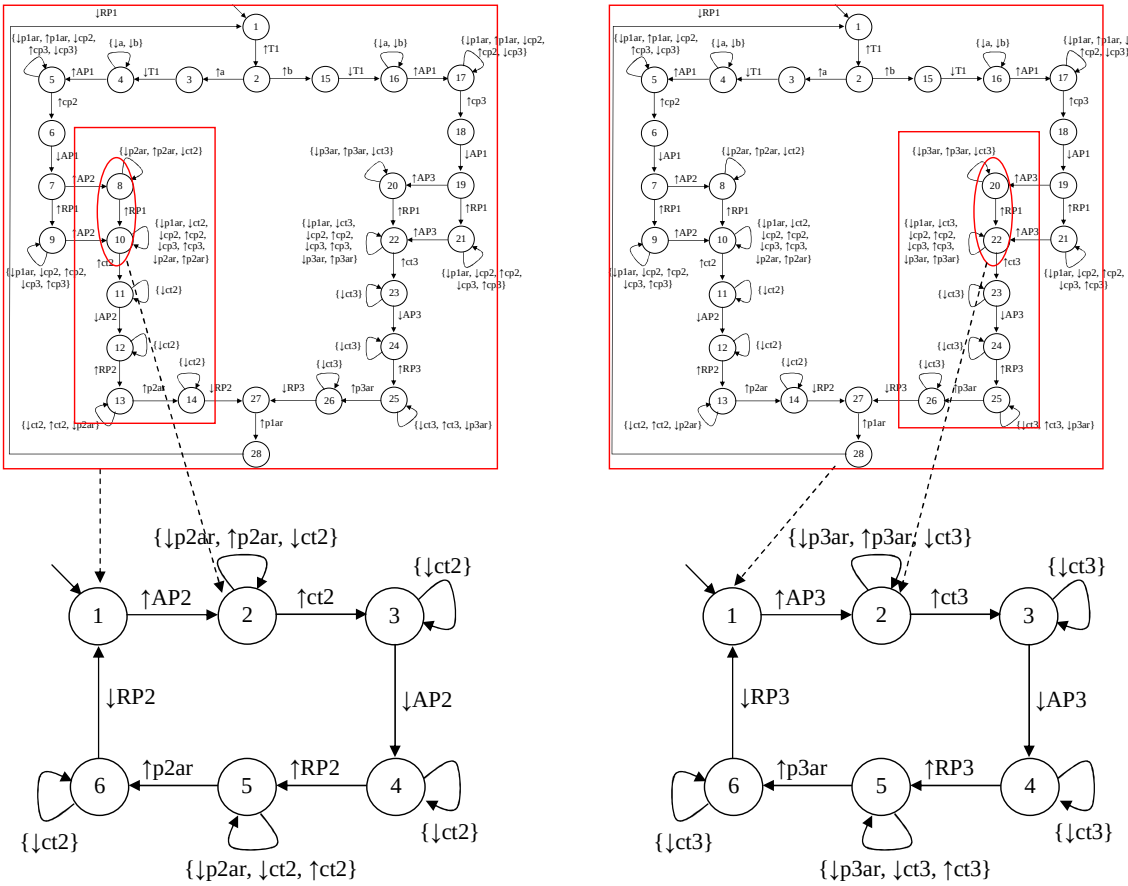
L'ensemble des événements du langage du poussoir 2 est {↓p2ar, ↑p2ar, ↓ct2, ↑ct2, ↓AP2, ↑AP2, ↓RP2, ↑RP2}. La restriction du langage conduit à un automate $GEREPO_{P2}$ à 6 états (Figure 4-25a) avec les regroupements du Tableau 4-14. Pour le poussoir 3, il en est de même avec un ensemble d'événements {↓p3ar, ↑p3ar, ↓ct3, ↑ct3, ↓AP3, ↑AP3, ↓RP3, ↑RP3} fournissant l'automate $GEREPO_{P3}$ de la Figure 4-25b avec les regroupements du Tableau 4-15.

Groupe 1 = {1, 2, 3, 4, 5, 6, 7, 9, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28}	Groupe 4 = {12}
Groupe 2 = {8, 10}	Groupe 5 = {13}
Groupe 3 = {11}	Groupe 6 = {14}

Tableau 4-14 : Regroupement pour le Graphe Equivalent Restreint du poussoir 2 : $GEREPO_{P2}$

Groupe 1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 27, 28}	Groupe 4 = {24}
Groupe 2 = {20, 22}	Groupe 5 = {25}
Groupe 3 = {23}	Groupe 6 = {26}

Tableau 4-15 : Regroupement pour le Graphe Equivalent Restreint du poussoir 3 : $GEREPO_{P3}$



a) GE Restreint pour le poussoir 2 $GEREPO_{P2}$ b) GE Restreint pour le poussoir 3 $GEREPO_{P3}$

Figure 4-25 : Graphe Equivalent Restreint pour le poussoir 2 et 3

4.3.5.3. Tapis 1

Pour le tapis 1, la restriction du langage porte sur les événements {↓a, ↑a, ↓b, ↑b, ↓T1, ↑T1}. L'automate $GEREPO_{T1}$ est composé de 4 états (Figure 4-26). Chacun des états 2, 3 et 15 du *GE* constitue un groupe qui va représenter les états 2, 3 et 4 du $GEREPO_{T1}$. L'état 1 de ce modèle est une agrégation d'un groupement des états {1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28}. On retrouve la possibilité d'avoir soit l'activation de *a* (état 3), soit l'activation de *b* (état 4) lorsque le tapis est en rotation (Tableau 4-16). Les boucles sur l'état 1 sont toujours sur la désactivation des détecteurs en attendant leur développement lors de l'étape de composition synchrone avec l'automate EPO_{T1} de la Figure 4-21c.

Groupe 1 = {1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28}	Groupe 3 = {3}
Groupe 2 = {2}	Groupe 4 = {15}

Tableau 4-16 : Regroupement pour le Graphe Equivalent Restreint du tapis 1 : $GEREPO_{T1}$

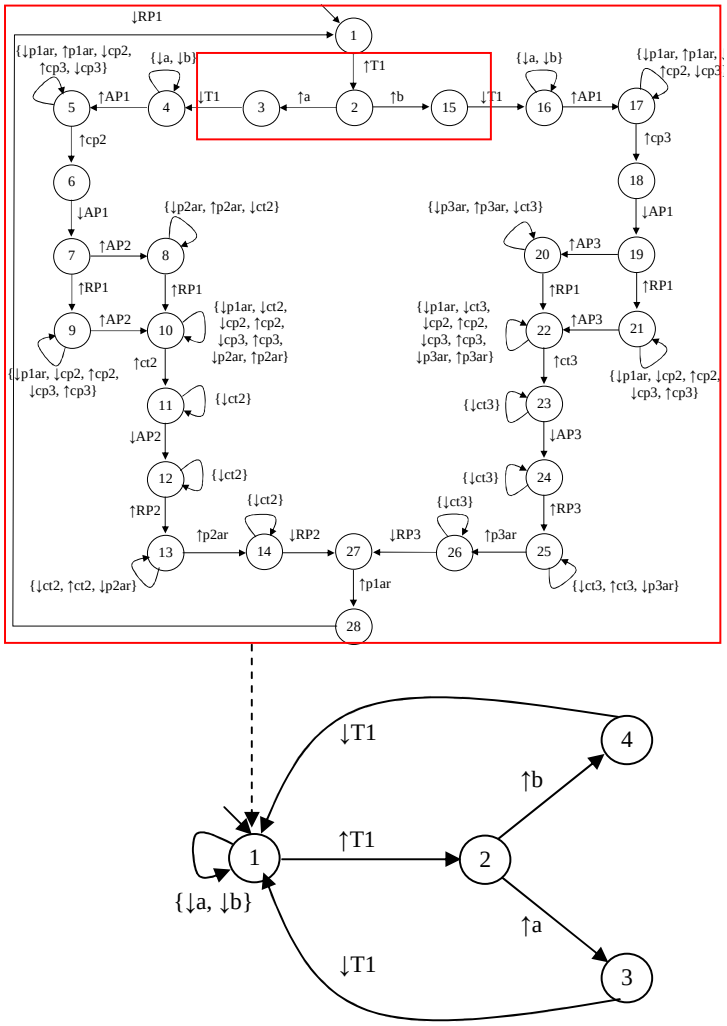


Figure 4-26 : Graphe Equivalent Restreint au langage de l'Elément de Partie Opérative du tapis 1 : GEREPO$_{T1}$

4.3.5.4. Tapis 2 et 3

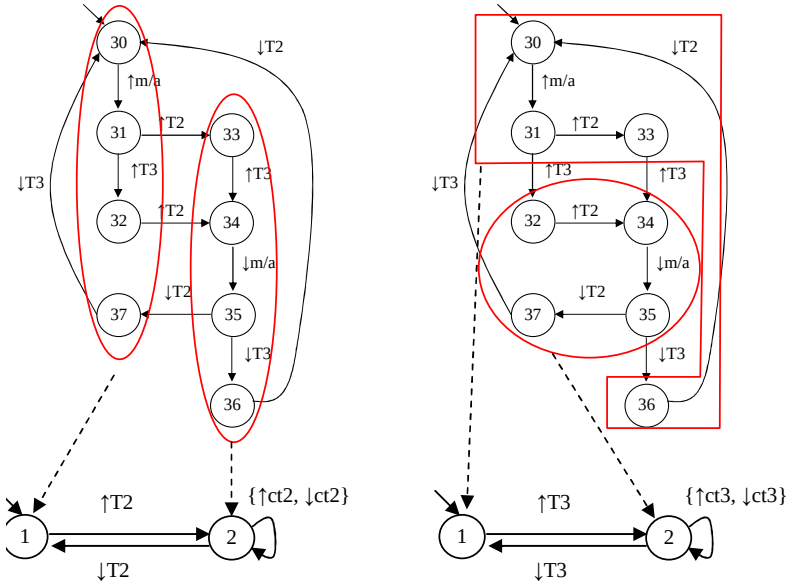
L'ensemble des événements du langage du tapis 2 est {↓ct2, ↑ct2, ↓T2, ↑T2} et du tapis 3 {↓ct3, ↑ct3, ↓T3, ↑T3}. Cependant, la restriction du langage ne s'effectue pas sur le *GE* du GRAFCET du système de tri de caisses (Figure 4-23a) mais sur celui de la gestion des tapis 2 et 3 (Figure 4-23b). Les états 30, 31, 32 et 37 du *GE* des tapis sont regroupés et agrégés en un état 1 du GEREPO$_{T2}$ (Tableau 4-17). Les états 33, 34, 35 et 36 sont, quant à eux, regroupés et agrégés dans l'état 2 du modèle GEREPO$_{T2}$ de la Figure 4-27a. L'automate GEREPO$_{T3}$ est construit de la même manière avec un premier groupement des états 30, 31, 33 et 36 (Tableau 4-18) et un second groupement des états 32, 34, 35 et 37 pour donner au final l'automate GEREPO$_{T3}$ de la Figure 4-27b.

Groupe 1 = {30, 31, 32, 37}	Groupe 2 = {33, 34, 35, 36}

Tableau 4-17 : Regroupement le Graphe Equivalent Restreint du tapis 2 : GEREPO$_{T2}$

Groupe 1 = {30, 31, 33, 36}	Groupe 2 = {32, 34, 35, 37}

Tableau 4-18 : Regroupement pour le Graphe Equivalent Restreint du tapis 3 : GEREPO$_{T3}$



a) $GEREPO_{T2}$ b) $GEREPO_{T3}$

Figure 4-27 : Graphe Equivalent Restreint pour le tapis 2 et 3

4.3.6. Obtention des EPOC

Les EPOC décrivant le comportement désiré de chaque EPO sont obtenus dans l'intersection par l'étape de composition synchrone entre l'automate *GE* restreint au langage désiré et l'automate d'EPO.

4.3.6.1. Elément de Partie Opérative Commandé du poussoir 1

La composition synchrone entre le modèle EPO_{P1} de la Figure 4-18c et l'automate $GEREPO_{P1}$ de la Figure 4-24 donne un automate $EPOC_{P1}$ décrivant le comportement désiré du poussoir 1 tel qu'il est décrit Figure 4-28. De l'état 1, il est possible d'avoir l'activation de l'ordre AP1 qui entraîne la désactivation du détecteur *p1ar* puis l'activation de *cp2* (état 4). Dans ce cas, il est possible soit de désactiver l'ordre AP1 et d'avoir l'ordre de retour RP1 si la caisse détectée est une petite caisse (états 13 et 14), soit de continuer l'avancée du poussoir par la désactivation de *cp2* jusqu'à l'activation de *cp3* dans le cas d'une grande caisse (états 5 et 6). Dans les deux situations, le retour du vérin engendre l'activation du détecteur de fin de course de rentrée *p1ar* (état 12) pour enfin désactiver l'ordre de retour RP1 et revenir en situation initiale.

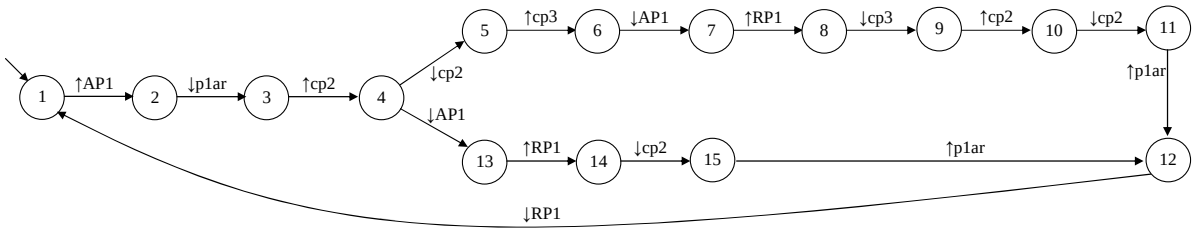


Figure 4-28 : Elément de Partie Opérative Commandé du poussoir 1 : $EPOC_{P1}$

4.3.6.2. Elément de Partie Opérative Commandé des poussoirs 2 et 3

La composition synchrone entre le modèle EPO_{P2} de la Figure 4-19c et l'automate $GEREPO_{P2}$ de la Figure 4-25a donne un automate $EPOC_{P2}$ à 13 états (Figure 4-29). A partir de l'état 1, il est possible d'activer l'ordre de sortie du poussoir AP2 qui engendre la désactivation de *p2ar* puis l'activation de *ct2* (état 4). Le détecteur *ct2* appartenant également au tapis, et décrivant le passage d'une caisse, implique que sa désactivation peut se produire à n'importe quel moment. Dès lors, l'automate possède une structure parallèle où il est possible de désactiver *ct2* après la désactivation de AP2 (état 5), l'activation de RP2 (état 6), jusqu'à l'activation du détecteur *p2ar* indiquant que le vérin est rentré (états 7). La désactivation de l'ordre RP2 engendre un retour en état initial directement si le détecteur *ct2* a bien été désactivé auparavant (état 13 puis 1). Sinon la désactivation de RP2 s'effectue avant la désactivation de *ct2* afin de redémarrer un cycle (état 8 puis 1). Le modèle $EPOC_{P3}$ est obtenu de la même manière et se présente sous la forme de l'automate de la Figure 4-30.

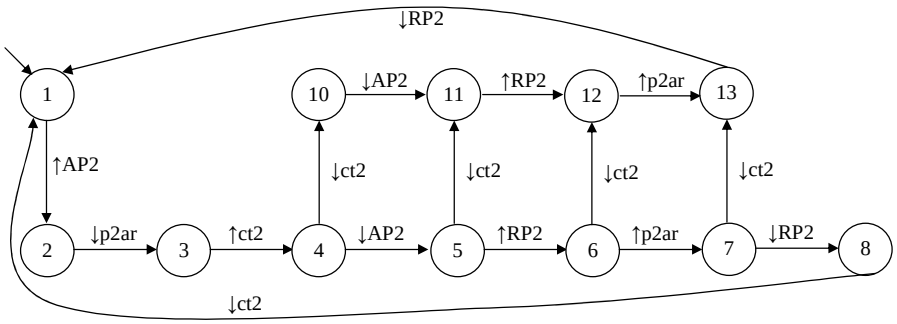


Figure 4-29 : Elément de Partie Opérative Commandé du poussoir 2 : $EPOC_{P2}$

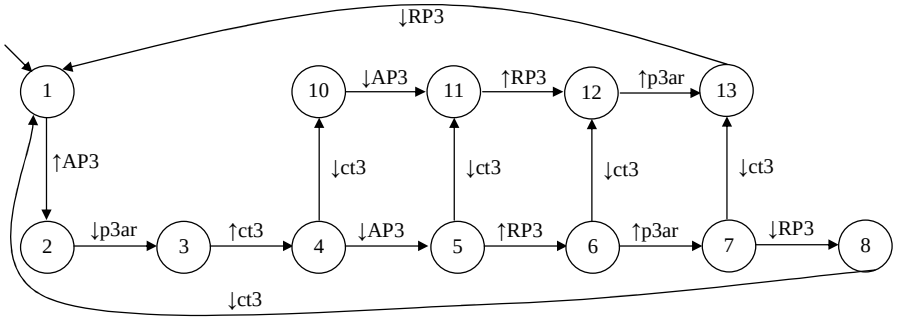


Figure 4-30 : Elément de Partie Opérative Commandé du poussoir 3 : $EPOC_{P3}$

4.3.6.3. Elément de Partie Opérative Commandé du tapis 1

L'automate $EPOC_{T1}$ du tapis 1 est obtenu par composition synchrone entre le modèle EPO_{T1} de la Figure 4-21c et l'automate $GEREPO_{T1}$ de la Figure 4-26. Le résultat de cette composition est l'automate de la Figure 4-31 où, à partir de l'état 1 et après activation de la rotation du tapis par l'ordre T1 (état 2), il est possible d'avoir, soit l'activation de *a* (état 3), soit l'activation de *b* (état 5). Dès lors, la désactivation du détecteur qui vient de s'activer ne peut se faire qu'après arrêt du moteur du tapis par la désactivation de l'ordre T1 (état 4 puis 1 pour *a* ou état 7 puis 1 pour *b*).

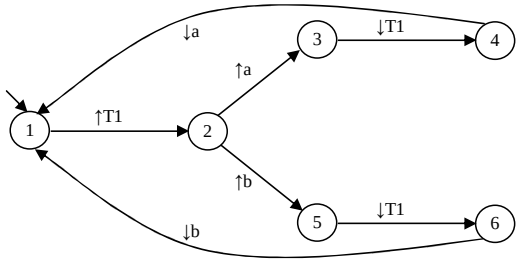
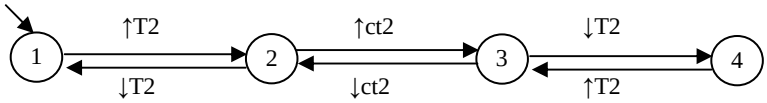


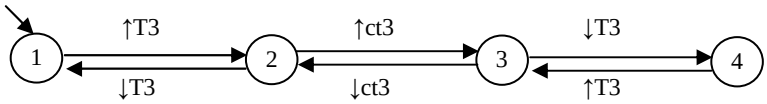
Figure 4-31 : Elément de Partie Opérative Commandé du tapis 1 : $EPOC_{T1}$

4.3.6.4. Elément de Partie Opérative Commandé des tapis 2 et 3

En ce qui concerne les tapis 2 et 3, les modèles $EPOC_{T2}$ et $EPOC_{T3}$ sont fournis en Figure 4-32. La composition synchrone du modèle EPO_{T2} (Figure 4-22c) avec celui du $GEREPO_{T2}$ (Figure 4-27a) aboutit à un modèle à 4 états identique à celui de l'EPO_{T2} (Figure 4-32a). En effet, le modèle $GEREPO_{T2}$ étant composé uniquement de 2 états pour l'ordre T2, il n'émet aucun comportement particulier qui soit différent de celui de l'EPO_{T2}. Le tapis T3 réagit de la même façon (Figure 4-32b).



a) $EPOC_{T2}$



b) $EPOC_{T3}$

Figure 4-32 : Elément de Partie Opérative Commandé du tapis 2 et 3 : $EPOC_{T2}$ et $EPOC_{T3}$

4.3.7. Fonctions de prévision

4.3.7.1. Information temporelle du poussoir 1

Prenons le poussoir 1, l'$EPOC_{P1}$ de la Figure 4-28 comporte 15 états représentant le comportement normal du poussoir 1. En ajoutant l'information du vecteur d'état et l'information temporelle, le modèle de l'$EPOC_{P1}$ est enrichi et désormais considéré comme un EPOC Temporel du poussoir 1 ($EPOCT_{P1}$) de la Figure 4-33. Un état normal de $EPOCT_{P1}$ est défini par son état x, son vecteur d'état $V_{P1}(x)$ = {p1ar, cp2, cp3, AP1, RP1} et l'ensemble des fonctions de prévision FP_x. Par exemple, l'état x_2 de l' $EPOCT_{P1}$ a pour vecteur d'état $V_{P1}(x_2)$ = (1 0 0 1 0) indiquant que le vérin poussoir 1 est en position de rentrée et que l'ordre de sortie vient d'être demandé. Une seule fonction de prévision est associée à cet état. Par conséquent, $FP_{x2} = FP(\uparrow AP1, \downarrow p1ar) = \{\uparrow AP1, x_2, (\downarrow p1ar, [1s, 2s], x_3, F1)\}$ est représentée en Figure 4-34 sur l'$EPOCT_{P1}$. Elle dépend de l'événement entrant ↑AP1 représentant l'événement déclencheur de la fonction de prévision et de l'événement sortant ↓p1ar qui doit arriver à l'état x_3 dans un intervalle défini [1s, 2s]. Si la fonction est respectée, le fonctionnement est considéré comme normal. Dans le cas contraire, $maxFP_{x2} = 1$, la fonction de prévision retourne une étiquette F1 correspondant à la partition Π_{F1}. Les fonctions de prévision sont définies pour chaque état lorsqu'un intervalle peut être donné. Elles permettent

ainsi de diminuer le retard de détection d'un défaut observable.

La grande difficulté est de récupérer les intervalles de temps pour chaque fonction. Ce travail demande soit une documentation technique riche sur l'EPO_i, soit un apprentissage à effectuer en amont sur la dynamique du procédé. Ces temps d'intervalle étant obtenus par apprentissage sur chaque composant, nous ne développerons pas chaque fonction de prévision dans ce paragraphe mais la Figure 4-33, représentant le comportement de l'$EPOCT_{P1}$ du diagnostiqueur D_{P1}, les intègre pour chaque état.

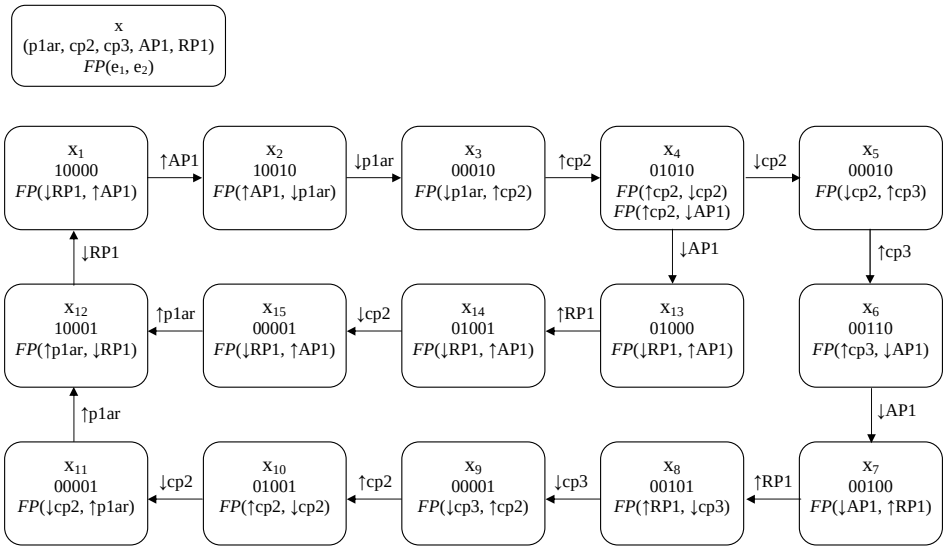


Figure 4-33 : EPOC Temporel ($EPOCT_{P1}$) du poussoir 1

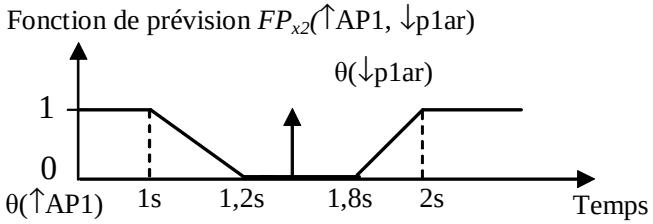


Figure 4-34 : Fonction de prévision de l'état x_2 de l'$EPOCT_{P1}$

4.3.7.2. Information temporelle des poussoirs 2 et 3

L'EPOC Temporel du poussoir 2 ($EPOCT_{P2}$) est représenté en Figure 4-35. Chaque état est associé à un vecteur d'état $V_{P2}(x)$ = (p2ar, ct2, AP2, RP2) et à un ensemble de fonctions de prévision FP_x. On remarque, par exemple, que l'état x_1 de la Figure 4-35 possède deux fonctions de prévision conditionnées par un événement déclencheur différent mais renvoyant la même étiquette de défaut. La première fonction de prévision est définie par *FP*(↓RP2, ↑AP2) = {↓RP2, x_1, (↑AP2, [1s, 3s], x_2, F9)} et la seconde par *FP*(↓ct2, ↑AP2) = {↓ct2, x_1, (↑AP2, [1s, 2s], x_2, F9)}. L'ensemble des fonctions de prévision est FP_{x1} = {*FP*(↓RP2, ↑AP2), *FP*(↓ct2, ↑AP2)}. Si l'une de ces fonctions n'est pas respectée, alors la valeur de la fonction de prévision est $maxFP_{x1}$ = 1 et l'étiquette retournée est celle de la fonction non respectée. Il s'agit alors d'un défaut lié à l'actionneur AP2 avec l'étiquette F9 pour l'état x_1. Il en est de même pour l'EPOC Temporel du poussoir 3 ($EPOCT_{P3}$) (Figure 4-36).

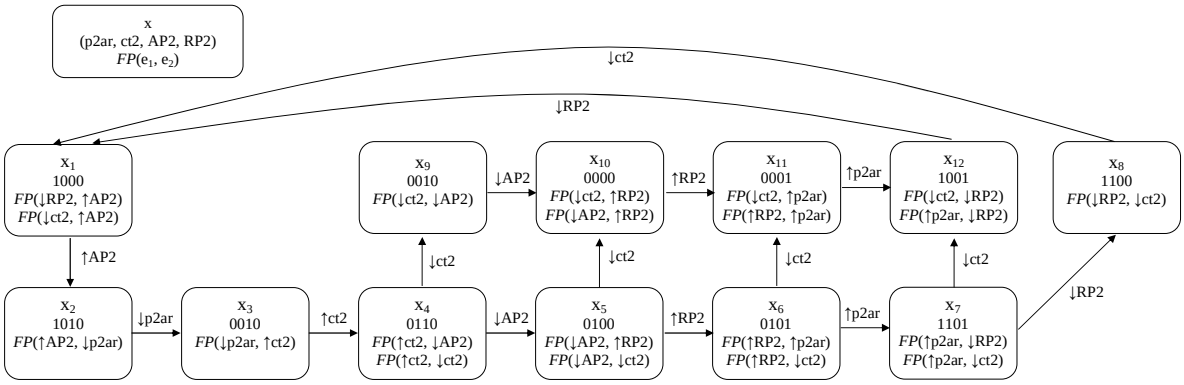


Figure 4-35 : EPOC Temporel ($EPOCT_{P2}$) du poussoir 2

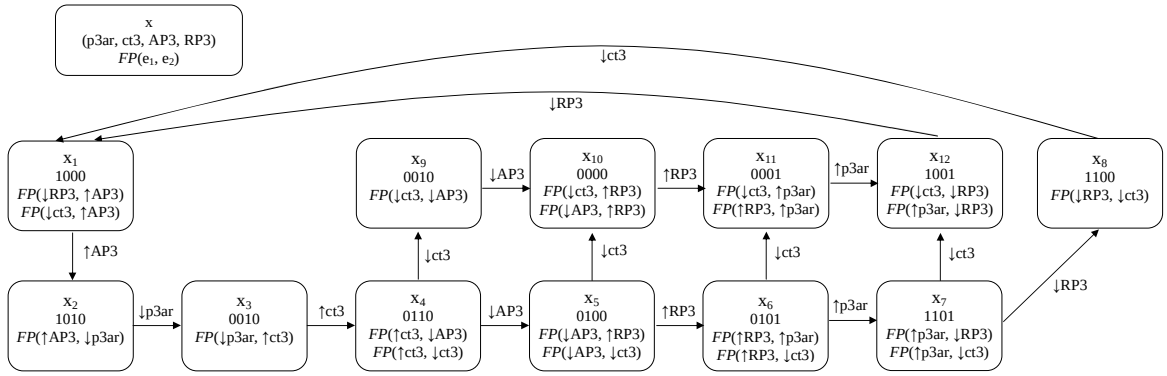


Figure 4-36 : EPOC Temporel ($EPOCT_{P3}$) du poussoir 3

4.3.7.3. Information temporelle du tapis 1

Pour le tapis 1, l'EPOC Temporel du tapis 1 ($EPOCT_{t1}$) est représenté en Figure 4-37. Comme pour les autres EPOCT, chaque état est associé à un vecteur d'état $V_{T1}(x)$ = (a, b, T1) et à un ensemble de fonctions de prévision FP_x. L'état x_2 de la Figure 4-37 possède deux fonctions de prévision conditionnées par le même événement déclencheur mais retournant deux étiquettes différentes. La première fonction de prévision est définie par FP(↑T1, ↑a) = {↑T1, x_2, (↑a, [1s, 2s], x_3, F17)} et retourne l'étiquette F17 en cas de violation de cette fonction de prévision. La seconde est FP(↑T1, ↑b) = {↑T1, x_2, (↑b, [1s, 2s], x_5, F18)} retourne l'étiquette F18 en cas de non respect de celle-ci. L'ensemble de ces fonctions de prévision est FP_{x2} = {FP(↑T1, ↑a), FP(↑T1, ↑b)}. Si l'une de ces fonctions n'est pas respectée, alors la valeur de la fonction de prévision est $maxFP_{x2}$ = 1 et l'étiquette retournée est celle de la fonction non respectée.

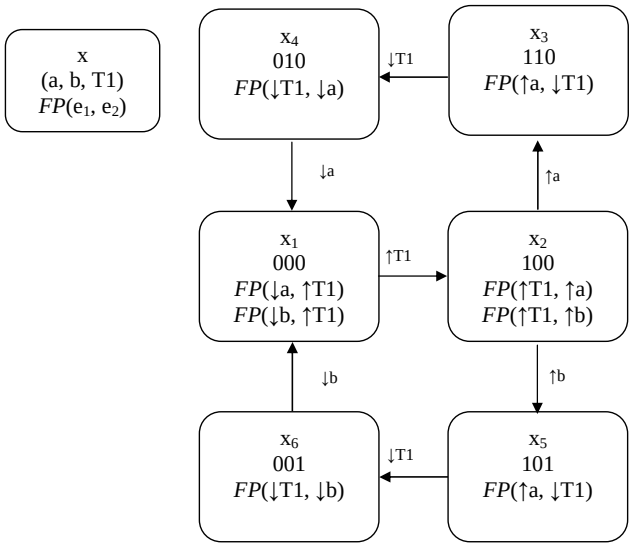


Figure 4-37 : EPOC Temporel (EPOCT$_{T1}$) du tapis 1

4.3.7.4. Information temporelle des tapis 2 et 3

Les EPOC Temporels des tapis 2 (EPOCT$_{T2}$) et 3 (EPOCT$_{T3}$) sont représentés respectivement en Figure 4-38 et Figure 4-39. Chaque état est également associé au vecteur d'état $V_{T2}(x)$ = (ct2, T2) pour l'EPOCT$_{T2}$ et au vecteur d'état $V_{T3}(x)$ = (ct3, T3) pour l'EPOCT$_{T3}$ avec un ensemble de fonctions de prévision FP_x. Il est à noter que l'état x_2 de la Figure 4-38, ou de la Figure 4-39, possède quatre fonctions de prévision conditionnées par deux événements déclencheurs et deux événements sortants. Elles représentent les possibilités d'évolutions temporelles entre les événements en fonction de la commande implantée par l'utilisateur. L'ensemble des fonctions de prévision de l'état x_2 de l'EPOCT$_{T2}$ est FP_{x2} = {*FP*(↑T2, ↑ct2), *FP*(↓ct2, ↓T2), *FP*(↑T2, ↓T2), *FP*(↓ct2, ↑ct2)} où les fonctions *FP*(↑T2, ↑ct2) et *FP*(↓ct2, ↑ct2) retournent une étiquette de défauts de type F8 et les fonctions *FP*(↓ct2, ↓T2) et *FP*(↑T2, ↓T2) retournent une étiquette de défauts de type F21. Si l'une de ces fonctions n'est pas respectée, alors la valeur de la fonction de prévision est $maxFP_{x2}$ = 1 et l'étiquette retournée est celle de la fonction non respectée.

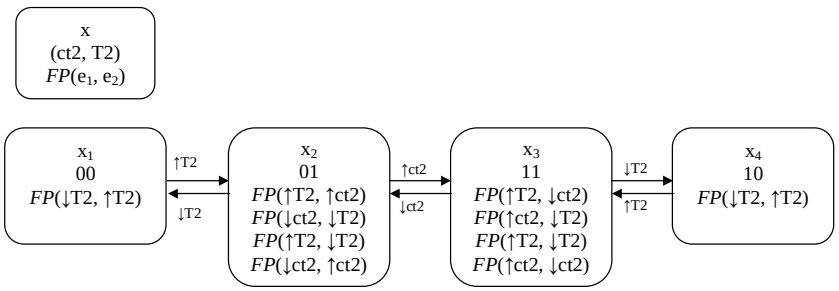


Figure 4-38 : EPOC Temporel (EPOCT$_{T2}$) du tapis 2

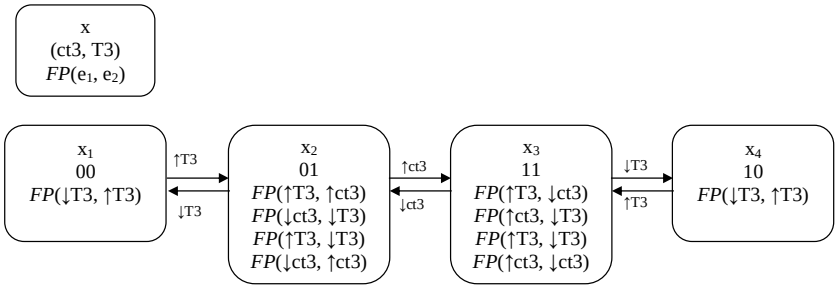


Figure 4-39 : EPOC Temporel ($EPOCT_{T3}$) du poussoir 3

4.3.8. Diagnostiqueurs locaux

Nous avions vu dans le chapitre 3 la création d'un EPOCT étendu avec un état de défaut atteint par l'occurrence d'un défaut. Ce modèle, permettant le passage de l'$EPOCT_i$ au diagnostiqueur local D_i, n'est cependant qu'une étape explicative du formalisme des diagnostiqueurs de l'approche de diagnostic décentralisé que nous proposons. C'est pourquoi nous ne développerons pas ces EPOCT étendus dans ce chapitre mais passerons directement à l'obtention des diagnostiqueurs locaux.

4.3.8.1. Diagnostiqueur du poussoir 1

Prenons l'exemple du poussoir 1 où le diagnostiqueur D_{P1} est obtenu à partir du comportement normal et de l'ensemble des défauts possibles décrit en Tableau 4-6. Ainsi, à partir de l'état x_1 avec le vecteur d'état $V_{P1}(x_1)$ = (p1ar, cp2, cp3, AP1, RP1) = (1 0 0 0 0), il est possible soit de diagnostiquer (Figure 4-40) :

- Un défaut f_{14} provenant de la PC est diagnostiqué par la fonction de prévision FP_{x1} qui retourne alors une étiquette F4 dans l'état de défaut x_{16},
- un défaut observable f_3 par l'observation d'un événement non attendu sur *p1ar* qui passe de la valeur 1 à 0. L'état x_{17} est alors associé à l'étiquette F1,
- un défaut observable f_5 par l'observation d'un événement non attendu sur *cp2* qui passe de la valeur 0 à 1 qui retourne alors une étiquette F2 dans l'état de défaut x_{18},
- un défaut observable f_9 par l'observation d'un événement non attendu sur *cp3* qui passe de la valeur 0 à 1. L'état x_{19} est alors associé à l'étiquette F3,
- un défaut observable f_{17} par sur l'ordre RP1 qui est activé par la commande alors qu'il n'aurait pas dû l'être et qui retourne alors une étiquette F5 dans l'état de défaut x_{20}.

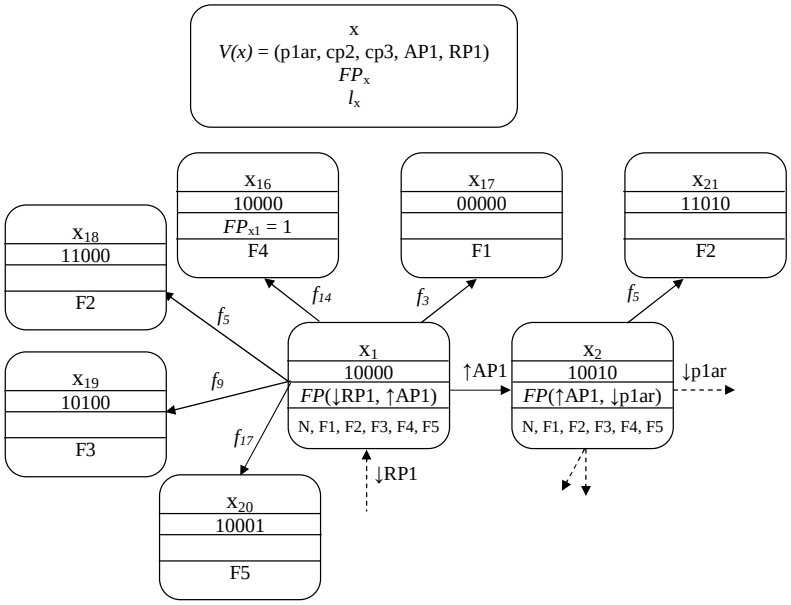


Figure 4-40 : Extrait du diagnostiqueur du poussoir 1 D_{P1}

Il est donc possible de diagnostiquer un défaut des 5 partitions possibles à partir de chaque état normal du diagnostiqueur D_{P1}. Pour une question de lisibilité, nous ne représenterons pas les 90 états du diagnostiqueur D_{P1}. Si l'on ne souhaite que représenter le diagnostiqueur simplifié aux défauts non observables mais diagnosticables, ce dernier possède alors que 30 états. Un extrait de ce diagnostiqueur est donné en Figure 4- 41.

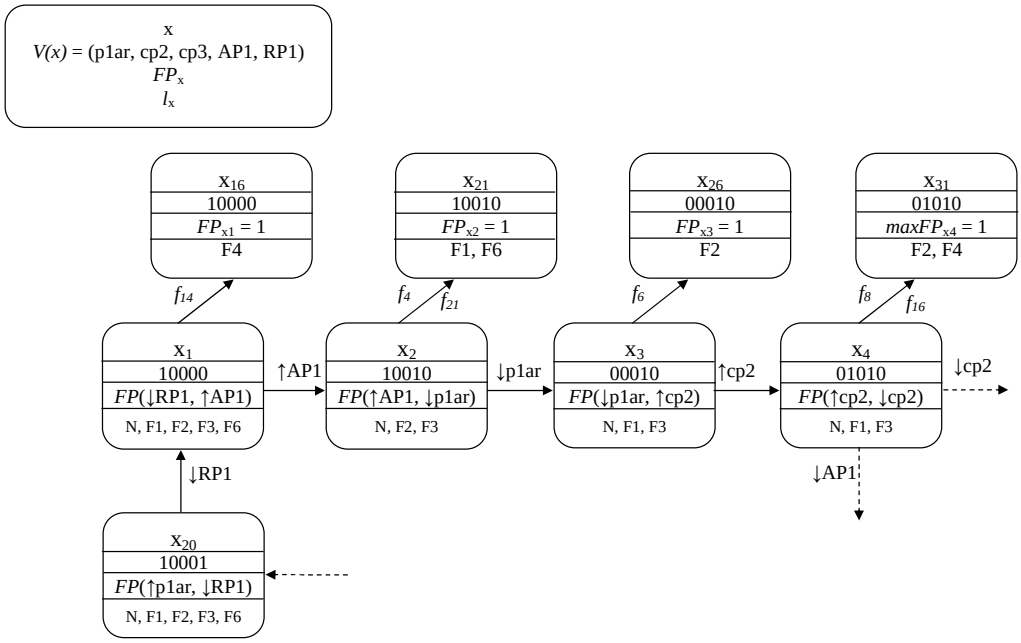


Figure 4- 41 : Extrait du diagnostiqueur du poussoir 1 D_{P1} simplifié aux défauts non observables

4.3.8.2. Complexité des différents diagnostiqueurs locaux

Pour les diagnostiqueurs locaux du poussoir 2 (D_{P2}) et du poussoir 3 (D_{P3}), il est possible, pour chaque état normal, de diagnostiquer l'ensemble des défauts observables et un défaut

non observable pour chaque fonction de prévision retournant une étiquette. Les diagnostiqueurs D_{P2} et D_{P3} possèdent chacun 60 états dont 48 sont associés à une étiquette de défauts permettant de diagnostiquer un défaut avec certitude. Pour une question de lisibilité, nous ne représenterons pas l'ensemble de ces diagnostiqueurs locaux dans ce chapitre. Nous pouvons cependant voir un récapitulatif de la complexité des diagnostiqueurs locaux dans le Tableau 4-19.

EPO	nbr. d'états normaux EPOC	nbr. de partitions possibles	nbr. d'états de défauts	nbr. d'états du diagnostiqueur	nbr. d'états du diagnostiqueur simplifié
P1	15	5	5 x 15 = 75	90	30
P2	12	4	4 x 12 = 48	60	24
P3	12	4	4 x 12 = 48	60	24
T1	6	3	3 x 6 = 18	24	12
T2	4	2	2 x 4 = 8	12	8
T3	4	2	2 x 4 = 8	12	8

Tableau 4-19 : Complexité des diagnostiqueurs locaux

A titre indicatif, un diagnostiqueur centralisé peut être obtenu par la composition synchrone entre l'ensemble des EPOC avec les deux *GE* des GRAFCET de commande. Après application de l'ensemble des 24 partitions de défauts à diagnostiquer, le diagnostiqueur global DG serait composé de 58 824 états. Nous voyons donc bien sur cet exemple l'intérêt de la décentralisation des composants.

4.3.9. Coordinateur

Nous venons d'établir l'ensemble des diagnostiqueurs locaux du comportement désiré de la PC. Il faut maintenant établir le coordinateur permettant la description des contraintes globales du système non définies par les diagnostiqueurs locaux et la gestion des cas d'indécision entre les différents éléments, et cela notamment pour le détecteur *ct2* commun au poussoir 2 et au tapis 2 ainsi que pour le détecteur *ct3* commun au poussoir 3 et au tapis 3. En effet, la partition de défauts Π_{F8}, concernant le détecteur *ct2* et retournant une étiquette F8, est présente au niveau des diagnostiqueurs locaux D_{P2} et D_{T2}. Il faut donc que ces deux diagnostiqueurs soient en accord pour chaque décision concernant cette partition. Si l'un de ces deux diagnostiqueurs diagnostique un défaut de type F8 et que l'autre diagnostiqueur ne garantit pas l'absence de ce défaut, NF8, alors il y a indécision. Dès lors, la décision finale sera l'occurrence d'un défaut de type F8. Il en est de même pour le détecteur *ct3* partagé entre les diagnostiqueurs locaux D_{P3} et D_{T3}.

4.3.9.1. Spécifications globales

Le coordinateur doit établir les contraintes de spécifications globales qui ne peuvent pas être définies par les diagnostiqueurs locaux. Chaque contrainte est exprimée par des règles qui doivent être respectées. L'ensemble des contraintes globales du coordinateur est rappelé dans le Tableau 4-20.

	Evénement	**Condition à respecter**	**Etiquette**
Règle 1	↑AP1	¬T1 = 1	F4
Règle 2	↑AP1	¬AP2 ∧ p2ar = 1	F4
Règle 3	↑AP1	¬AP3 ∧ p3ar = 1	F4
Règle 4	↑AP2	¬AP1 ∧ cp2 ∧ ¬cp3= 1	F9
Règle 5	↑AP3	¬AP1 ∧ cp3 = 1	F14
Règle 6	↑ct2	T2 ∧ AP2 = 1	F8
Règle 7	↑ct3	T3 ∧ AP3 = 1	F13

Tableau 4-20 : Règles de contraintes globales du coordinateur C

La règle 1 exprime une contrainte entre le poussoir 1 et le tapis 1. En effet, il faut vérifier que le tapis 1 soit arrêté pour autoriser l'activation de la sortie du vérin poussoir 1. La règle 2 exprime la contrainte qui doit respecter le fait que l'autorisation de la sortie du vérin poussoir 1 n'est possible que si le poussoir 2 n'est pas activé et qu'il est en position de rentrée *p2ar*. La règle 3 exprime la même contrainte mais cette fois-ci entre le poussoir 1 et le poussoir 3. La règle 4, respectivement 5, autorise l'activation de la sortie du vérin poussoir 2, respectivement 3, uniquement si une caisse est en *cp2*, respectivement *cp3*, et que si la sortie du poussoir 1 n'est plus activée. Enfin, la règle 6 représente la contrainte sur le détecteur commun *ct2* entre le tapis 2 et le poussoir 2. Le détecteur *ct2* doit s'activer uniquement lorsque le tapis 2 est en marche et que le vérin poussoir 2 a son ordre de sortie activé. Il en est de même pour la règle 7 sur le détecteur *ct3* commun au poussoir 3 et au tapis 3. Une règle non respectée engendre une étiquette correspondant au défaut sur l'événement. Ainsi, si les règles 1, 2 et 3 ne sont pas respectées, alors l'étiquette retournée est F4. Il en est de même pour les autres règles.

4.3.9.2. Table de décisions

Le coordinateur doit également s'approprier l'ensemble des décisions locales des 6 diagnostiqueurs locaux avec l'ensemble des règles globales qu'il contient afin de retourner une décision globale sur le procédé tout en gérant les ambiguïtés et les cas d'indécision. Un extrait de la table de décision finale est donné dans le Tableau 4-21 :

- Le cas 1 montre une ambiguïté de décision sur l'ensemble des diagnostiqueurs locaux. Le coordinateur garantissant l'absence des défauts (NF4, NF8, NF9, NF13 et NF14), il est possible alors de donner une décision finale équivalente à celle du coordinateur.
- Le cas 2 indique un défaut de type F14 sur le diagnostiqueur du poussoir 3, D_{P3}, en accord avec la décision du coordinateur. La décision finale est donc un défaut de type F14.
- Le cas 3 représente un accord de décision sur la partition qui est en commun entre le diagnostiqueur D_{P2} et D_{T2}. Les deux diagnostiqueurs retournent bien l'étiquette F8 qui permet de conclure à un défaut de type F8 pour la décision finale.
- Le cas 4 indique un défaut de type F8 sur le détecteur commun *ct2* par le diagnostiqueur D_{P2}. Bien que le diagnostiqueur D_{T2} ne détecte pas ce défaut, la

décision finale est de prendre en compte l'étiquette en défaut F8.

- Le cas 5 est une ambiguïté de décision sur l'ensemble des diagnostiqueurs mais avec une détection d'un défaut de type F8 par le coordinateur. La décision finale est donc ce défaut de type F8. Ce cas d'indécision peut survenir puisque le coordinateur exprime les interactions entre les EPO que les diagnostiqueurs locaux ne peuvent pas voir seuls.
- Le cas 6 indique un défaut de type F4 par le diagnostiqueur D_{P1} et un défaut de type F13 par le diagnostiqueur D_{T3}. La décision du coordinateur sur le défaut de type F4 est donc inutile. La décision finale retourne donc un défaut multiple de type F4F13.

Cas	DD_{P1}	DD_{P2}	DD_{P3}	DD_{T1}	DD_{T2}	DD_{T3}	Décision du Coordinateur DC	Décision Finale DF
1	-	-	-	-	-	-	NF4, NF8, NF9, NF13, NF14	DC
2	-	-	F14	-	-	-	**F14**, NF4, NF8, NF9, NF13	DC
3	-	F8	-	-	F8	-	NF4, NF9, NF13, NF14	F8, NF4, NF9, NF13, NF14
4	-	F8	-	-	-	-	NF4, NF9, NF13, NF14	F8, NF4, NF9, NF13, NF14
5	-	-	-	-	-	-	**F8**, NF4, NF9, NF13, NF14	DC
6	F4	-	-	-	-	F13	NF8, NF9, NF14	F4, F13, NF8, NF9, NF14
…	…	…	…	…	…	…	…	…

Tableau 4-21 : Extrait de la table de décision finale

4.3.10. Conclusion et vérification des performances

L'application de notre démarche sur le système manufacturier du tri de caisses montre bien la nécessité de disposer d'une bibliothèque d'EPO afin d'extraire au mieux le comportement désiré de chaque élément localement. Cet exemple montre également que le cas de détecteurs communs entre EPO n'empêche en rien la création de diagnostiqueurs locaux. La décision finale est alors prise au niveau du coordinateur devant gérer les conflits.

L'établissement des règles, exprimant des contraintes globales, dans le coordinateur n'est pas toujours exhaustif car rien ne permet d'assurer d'avoir établi l'ensemble de ces contraintes. Cependant, il est possible de vérifier si l'ensemble des défauts est diagnosticable. Pour cela, il faut passer par la notion de co-diagnosticabilité mixte développée dans le chapitre 3. La notion de co-diagnosticabilité mixte, à base d'événements, d'états et temporelle, permet de vérifier que l'ensemble des partitions de défauts diagnosticables par une structure centralisée est également co-diagnosticable par les diagnostiqueurs locaux de la structure décentralisée avec coordinateur.

Les performances sur la co-diagnosticabilité des défauts de l'approche de diagnostic décentralisée avec coordinateur sont maintenues. En effet, un diagnostiqueur global D_G avec une structure centralisée permet de diagnostiquer l'ensemble des partitions $\Sigma_{\Pi DG} = \{\Pi_{F1}, \Pi_{F2}, \Pi_{F3}, \Pi_{F4}, \Pi_{F5}, \Pi_{F6}, \Pi_{F7}, \Pi_{F8}, \Pi_{F9}, \Pi_{F10}, \Pi_{F11}, \Pi_{F12}, \Pi_{F13}, \Pi_{F14}, \Pi_{F15}, \Pi_{F16}, \Pi_{F17}, \Pi_{F18}, \Pi_{F19}, \Pi_{F20}, \Pi_{F21}, \Pi_{F22}, \Pi_{F23}, \Pi_{F24}\}$. Maintenant, si nous regardons l'ensemble des partitions des diagnostiqueurs locaux, alors $\Sigma_{\Pi DP1} \cup \Sigma_{\Pi DP2} \cup \Sigma_{\Pi DP3} \cup \Sigma_{\Pi DT1} \cup \Sigma_{\Pi Dt2} \cup \Sigma_{\Pi DT3} = \Sigma_{\Pi DG}$. Par conséquent, nous sommes capables de diagnostiquer l'ensemble des partitions qu'un diagnostiqueur global peut diagnostiquer.

Les matrices de défauts, définies pour chaque boucle des diagnostiqueurs locaux, permettent de donner le retard de détection maximum. Ainsi, le Tableau 4- 22 présente la matrice de défauts du diagnostiqueur D_{P1} pour la boucle d'états (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}). Cette matrice indique que le retard de détection maximum est de 8 événements observables, $RD_{1P1max} = 8$.

	Etat / Défaut	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
F1	f_2	1	1	0	0	0	0	0	0	0	0	1	1
	f_4	0	1	1	1	1	1	1	1	1	1	1	0
F2	f_6	0	0	1	1	0	0	0	0	1	1	0	0
	f_8	1	1	1	1	1	1	1	1	1	0	0	0
F3	f_{10}	0	0	0	0	1	1	1	1	0	0	0	0
	f_{12}	1	1	1	1	1	0	0	1	1	1	1	1
F4	f_{14}	1	1	1	1	1	1	1	1	1	1	1	1
	f_{16}	1	1	1	1	1	1	1	1	1	1	1	1
F5	f_{18}	1	1	1	1	1	1	1	1	1	1	1	1
	f_{20}	1	1	1	1	1	1	1	1	1	1	1	1
F6	f_{21}	0	1	1	1	1	1	1	1	1	1	1	0
	f_{22}	1	1	1	1	1	0	0	1	1	1	1	1

Tableau 4- 22 : Matrice de défauts d'une boucle du diagnostiqueur D_{P1}

En regardant l'ensemble de ces matrices de défauts, il est possible de connaître le retard de détection sur l'ensemble du procédé. La notion de co-diagnosticabilité mixte permet donc de vérifier nos capacités au diagnostic avec un retard de détection maximal de 8 événements observables.

Dans le cas où une partition de défauts diagnosticable par la structure centralisée ne serait pas co-diagnosticable par la structure décentralisée, il faudrait alors établir de nouvelles règles permettant le diagnostic de cette partition dans le coordinateur afin de garantir les mêmes performances.

4.4. Simulation de l'approche de diagnostic décentralisé avec coordinateur

Nous avons vu comment obtenir les diagnostiqueurs locaux d'un procédé à partir de ses éléments. L'étape suivante, de l'approche développée dans cette thèse, est normalement l'étape d'implantation de l'approche dans un filtre intercalé entre la PC et la PO. Cependant, nous souhaitons tester notre approche avant de l'implanter. Nous avons donc créé un outil de simulation basé sur un simulateur réagissant à des signaux binaires et permettant le diagnostic de défauts. Les diagnostiqueurs étant modélisés par des automates à états évolués, il est possible de les considérer comme des *Statecharts*. Nous avons opté pour l'outil de simulation *Stateflow* du logiciel MatLab/Simulink [MAT 04], [ROU 04]. *Stateflow* est un outil de développement et de conception graphique puissant. Il permet notamment de visualiser et de simuler les systèmes réactifs complexes basés sur la théorie des machines à états finis. Destiné à la base pour représenter les *Statecharts*, nous l'avons utilisé dans le cadre de nos applications pour la simulation des diagnostiqueurs.

4.4.1. Application à l'exemple du wagonnet

Pour illustrer nos propos, nous avons choisi de reprendre l'exemple du wagonnet de la Figure 4-1 composé d'un vérin et d'un chariot en déplacement. Les différentes partitions de défauts sont rappelées dans le Tableau 4-1.

La simulation doit reprendre les deux diagnostiqueurs locaux ainsi que le coordinateur exprimant les contraintes globales du système. La simulation du wagonnet est représentée en Figure 4-42. Elle présente deux diagnostiqueurs correspondant aux deux éléments du système que sont le vérin double effet et le chariot. Pour simuler le comportement de la PC et de la PO, nous avons placé deux modules composés d'une série d'interrupteurs permettant la simulation des entrées/sorties des éléments.

Les diagnostiqueurs locaux retournent, à la table des décisions du coordinateur, leur décision par des étiquettes de défauts ainsi que l'étiquette correspondant à la fonction de prévision non respectée. Le coordinateur comprend, quant à lui, un module gérant les règles des contraintes globales ne pouvant être exprimées par les diagnostiqueurs locaux, comme les interactions entre les composants, permettant de retourner une étiquette à la table des décisions. L'agrégation de l'ensemble des décisions locales et des règles du coordinateur permet alors de rendre une décision finale sur le comportement du système à l'utilisateur. Nous allons voir chaque module (diagnostiqueurs locaux et coordinateur) en détail.

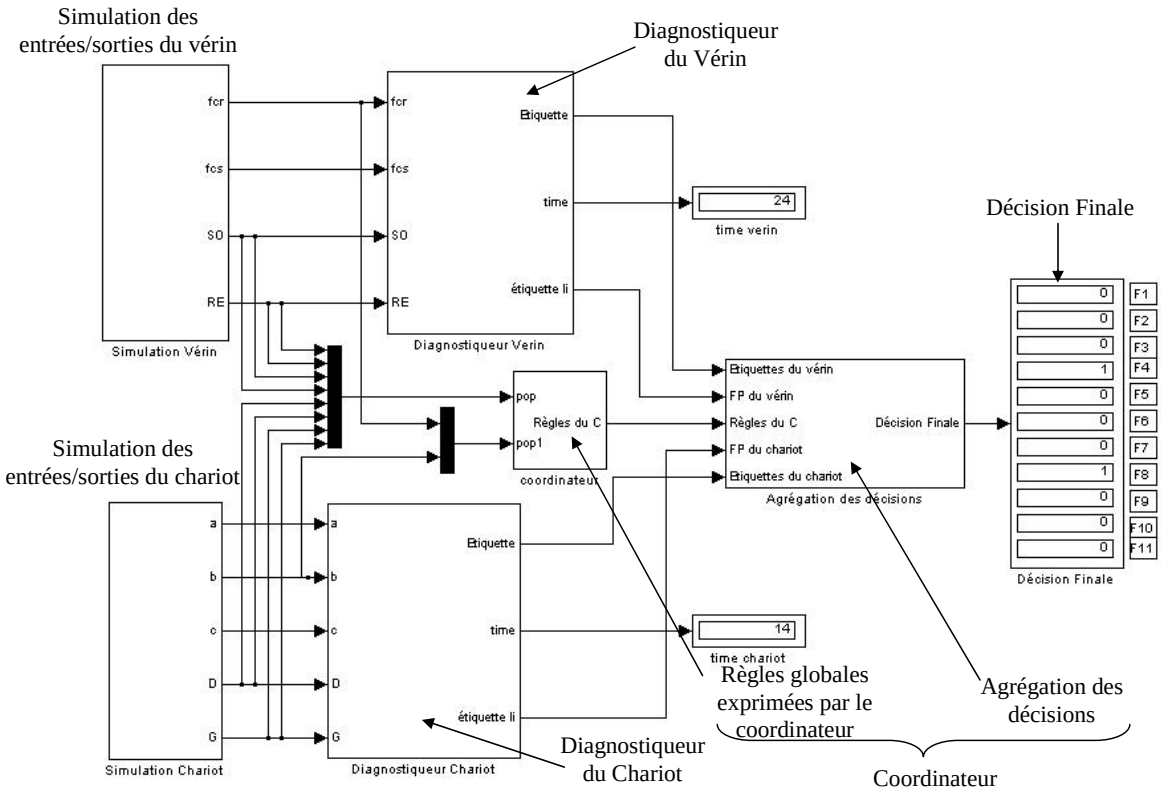


Figure 4-42 : Simulation du procédé du wagonnet de la Figure 4-1

4.4.2. Diagnostiqueur local du vérin

4.4.2.1. Module du diagnostiqueur local du vérin

Le module du vérin est présenté Figure 4-43. Il est composé d'un générateur d'impulsions représentant l'horloge interne du procédé pour la simulation, d'un module comportant le diagnostiqueur du vérin simplifié aux défauts non observables, d'un opérateur logique "OU exclusif" et d'un historique des événements.

L'ensemble des événements, provenant de la PC ou de la PO, attaque le module diagnostiqueur qui doit répondre par la concordance ou non des événements attendus. La détection des défauts observables se fait implicitement par l'opérateur logique "OU exclusif" permettant de visualiser le dernier événement observable arrivé sur le diagnostiqueur et de le comparer avec l'événement attendu. Ainsi, si l'événement réel ne correspond pas à l'événement attendu en sortie du module diagnostiqueur, alors un défaut observable est détecté et localisé par un 1 logique sur la variable d'état du vecteur d'état $V_1(x)$. Par contre, les défauts non observables devant être détectés par une fonction de prévision, sont représentés dans le module diagnostiqueur du vérin.

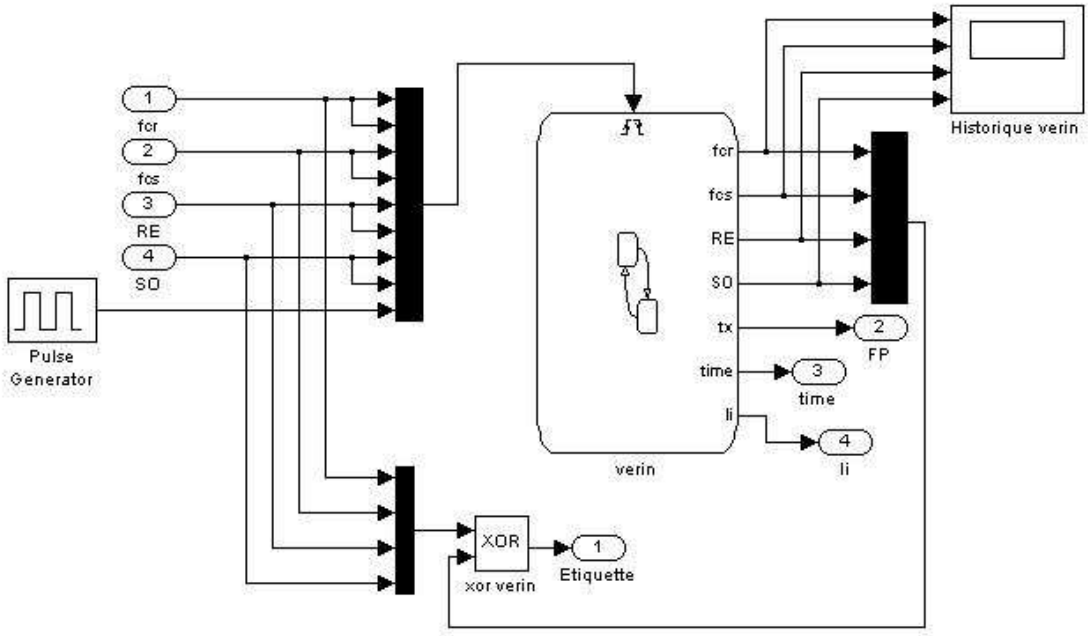


Figure 4-43 : Module du diagnostiqueur D_1 du vérin double effet avec ses entrées/sorties

4.4.2.2. Diagnostiqueur local du vérin simplifié

Le diagnostiqueur D_1 du vérin est celui de la Figure 4-15 simplifié aux défauts non observables détectés à travers les fonctions de prévision. La Figure 4-44 présente le diagnostiqueur D_1 simplifié aux défauts non observables où l'état initial est représenté par une flèche avec un point. *Stateflow*, ne gérant pas les symboles ↓ et ↑, le front montant d'un événement est représenté par les lettres "fm" devant l'événement alors que le front descendant est décrit par les lettres "fd".

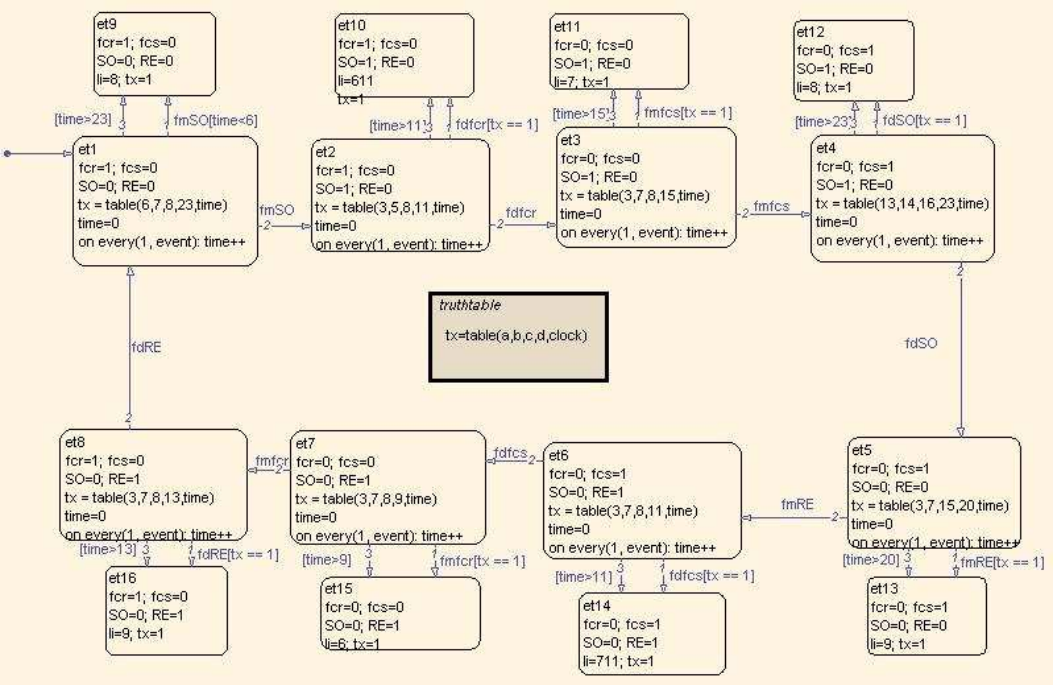


Figure 4-44 : Diagnostiqueur D_1 du vérin simplifié aux défauts non observables

4.4.2.3. Détails d'un état du diagnostiqueur local

Pour cette simulation, le diagnostiqueur D_1 du vérin comporte 16 états. Chaque état normal (1, 2, 3, 4, 5, 6, 7, 8) est composé d'un numéro d'état, du vecteur d'état $V_1(x)$ = {fcr, fcs, SO, RE} et d'une fonction de prévision (Figure 4-45). La fonction de prévision est établie selon une table de vérité *FP* = tx = table(a, b, c, d, clock), d'une variable temps, *time*, initialisée par l'arrivée d'un événement sur l'état et d'une fonction d'incrémentation du temps "*on every*" selon une base de temps *event*. Ainsi, à chaque fois que la base de temps, *event*, définie par l'horloge, occurre, la variable de temps, *time*, de l'état courant est incrémenté. En retour, la table de vérité renvoie la valeur correspondante du temps. Les états de défauts (9, 10, 11, 12, 13, 14, 15, 16) possèdent un vecteur d'état $V_1(x)$ mais également le numéro de l'étiquette à renvoyer, li, et la valeur de la fonction de prévision FP_x = tx.

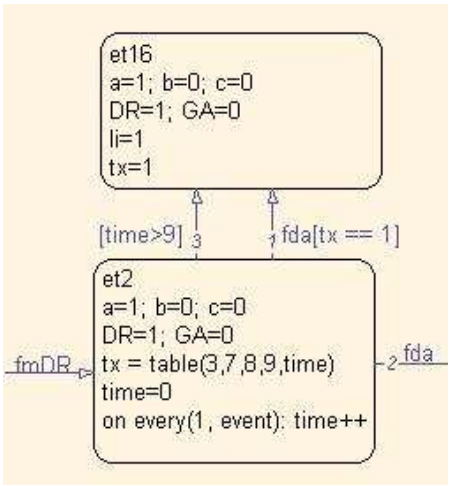


Figure 4-45 : Extrait montrant un état normal et un état défaillant du diagnostiqueur du vérin

4.4.2.4. Etablissement des fonctions de prévision

Les paramètres d'intervalles de temps de la table de vérité peuvent être modifiés pour chaque état. En effet, la table de vérité est définie par *FP* = tx = table(a, b, c, d, time). L'allure de la fonction de prévision est donnée par une table de conditions (Figure 4-46) et une table d'actions (Figure 4-47). La table de conditions décrit les actions à effectuer dans les intervalles de temps. Ainsi, si la variable *time* est inférieure ou égale au temps *clock* = a, alors il faut exécuter l'action A1. La table d'actions doit par conséquent renvoyer la valeur *FP* = 1. Il faut établir 5 conditions dans la table correspondant à chaque zone d'évolution de la fonction de prévision qui répondront à 5 actions dans la table d'actions.

Condition Table:

	Description	Condition	D1	D2	D3	D4	D5	D6
1	clock [0,a]	EQ1: clock <= a	T	F	F	F	F	-
2	clock]a,b[	EQ2: clock > a & clock < b	F	T	F	F	F	-
3	clock [b,c]	EQ3: clock >= b & clock <= c	F	F	T	F	F	-
4	clock]c,d[	EQ4: clock > c & clock < d	F	F	F	T	F	-
5	clock [d,+inf[	EQ5: clock >= d	F	F	F	F	T	-
		Actions: Specify a row from the Action Table	A1	A2	A3	A4	A5	A6

Figure 4-46 : Table de conditions de la fonction de prévision

Action Table:

#	Description	Action
1	Decision 1	A1: FP = 1;
2	Decision 2	A2: FP = (clock - b)/(a - b);
3	Decision 3	A3: FP = 0;
4	Decision 4	A4: FP = (clock - c)/(d - c);
5	Decision 5	A5: FP = 1;
6	Decision 6	A6: FP = 1;

Figure 4-47 : Table d'actions de la fonction de prévision

4.4.3. Diagnostiqueur local du chariot

En ce qui concerne le module du chariot, il possède également un diagnostiqueur et un historique des événements (Figure 4-48). Le générateur d'impulsions représente toujours l'horloge interne qui est enclenchée simultanément pour chaque diagnostiqueur. Le diagnostiqueur D_2 du chariot simplifié aux défauts non observables est représenté partiellement avec les états de défauts non observables par la Figure 4-49 où chaque état du diagnostiqueur possède également un numéro d'état, un vecteur d'état $V_2(x)$ = {a, b, c, DR, GA} et d'une fonction de prévision.

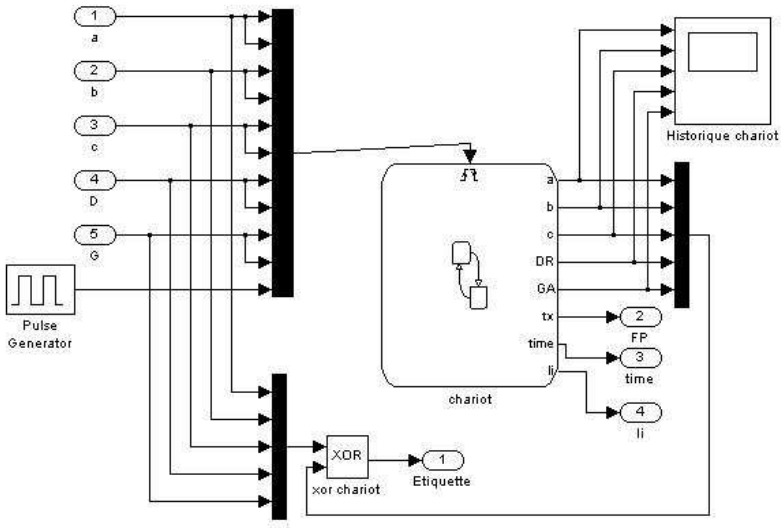


Figure 4-48 : Module du diagnostiqueur D_2 du chariot avec ces entrées/sorties

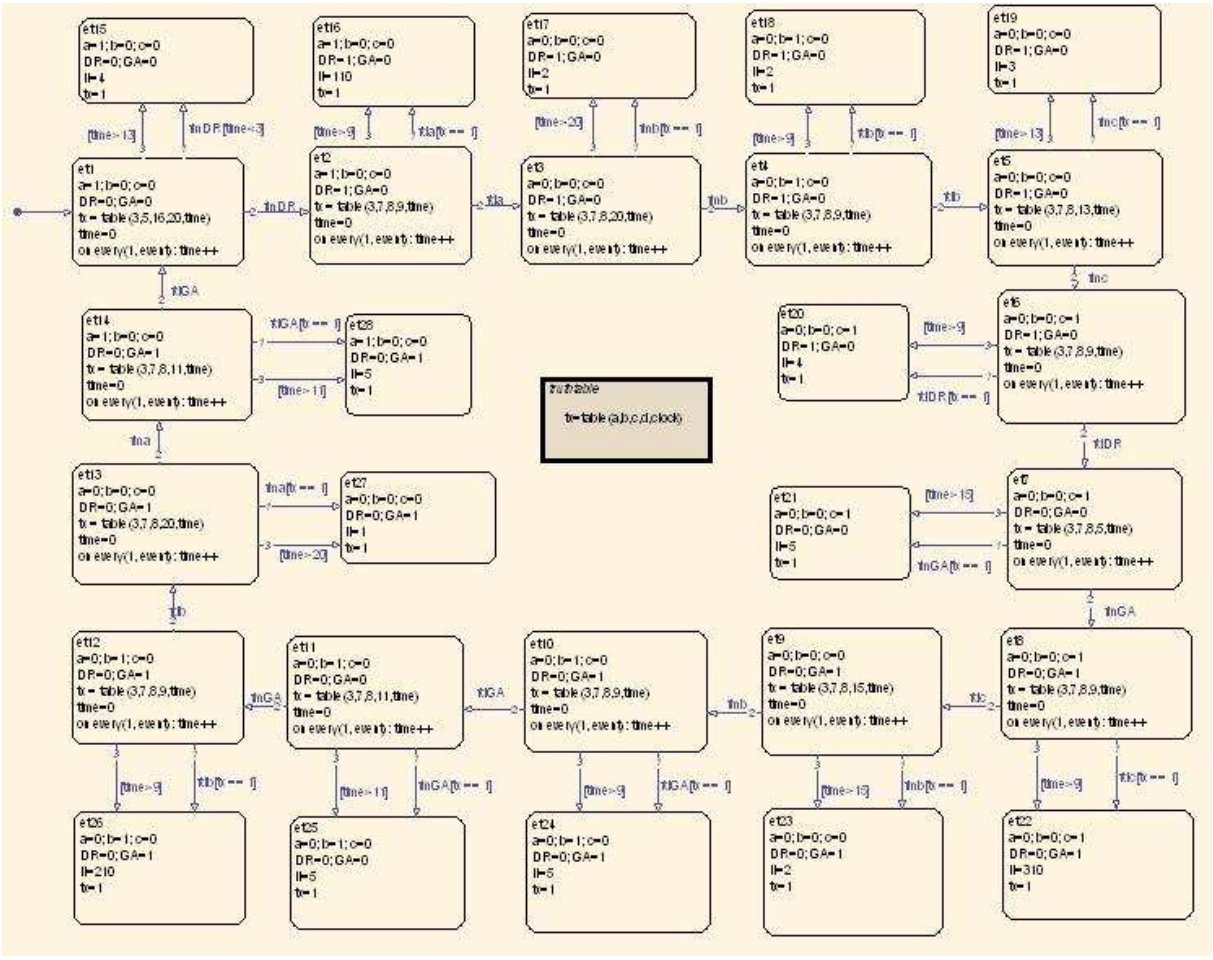


Figure 4-49 : Diagnostiqueur D_2 du chariot simplifié aux défauts non observables

4.4.4. Le coordinateur

Chaque diagnostiqueur local prend sa décision localement et renseigne l'opérateur à travers la valeur de la fonction de prévision de l'état en cours, l'historique des événements, le vecteur d'état et le dernier événement survenu sur l'EPO. Il ne reste plus qu'à exprimer les spécifications globales du procédé ne pouvant pas être décrites par les diagnostiqueurs locaux. Ce sont les interactions entre éléments qui doivent être établies à travers des règles dans le coordinateur. Nous avions vu que celui-ci décrivait essentiellement quatre contraintes rappelées dans le Tableau 4-2.

L'ensemble des règles s'exprime dans un module coordinateur *Stateflow* par des automates (Figure 4-42). Les règles 1 et 2 possèdent les mêmes conditions à respecter. Il est donc possible de les représenter par un automate à 3 états attendant en entrée, à partir de l'état *et1*, les événements ↑SO (état *et2*) et ↑RE (état *et3*) des règles. En sortie, l'automate répond par la condition à respecter sur *b*, D et G par rapport à la situation réelle du procédé. Par exemple, la règle 1 est décrite par le passage de l'état 1 à l'état 2 du coordinateur de la Figure 4-50 où l'activation de SO doit respecter les valeurs de $b = 1$, DR = 0 et GA = 0. Dans le cas contraire, le coordinateur renvoie l'étiquette du défaut correspondant à l'utilisateur. Il en est de même pour la description des règles 3 et 4, dans un automate à 3 états, conditionnées par les mêmes valeurs sur *fcr*, *SO* et *RE*.

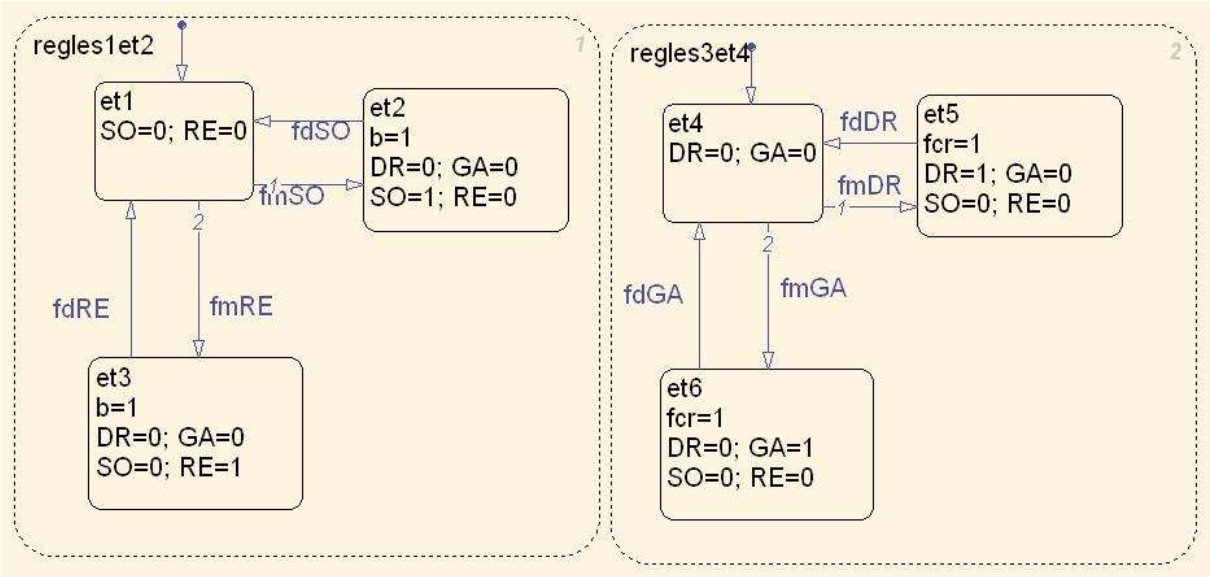


Figure 4-50 : Automate des règles du coordinateur

Le coordinateur doit également récupérer l'ensemble des décisions locales afin de rendre une décision finale à l'utilisateur. Pour cela, nous avons créé un module d'agrégation des décisions collectant l'ensemble des étiquettes de défauts des diagnostiqueurs locaux correspondant soit à un défaut observable (Etiquettes du vérin, Etiquettes du chariot), soit à un défaut non observable détecté par une fonction de prévision (FP du vérin, FP du chariot), mais également les étiquettes de défauts provenant des règles globales du coordinateur non respectées (Figure 4-51). Le module regarde ainsi l'ensemble des décisions qui arrive et envoie, après agrégation, la décision finale, sur l'état de fonctionnement du système, à l'utilisateur. Dans l'exemple du wagonnet, la décision finale est obtenue par un "1" logique à côté de chaque étiquette, de F1 à F11, correspondant à un défaut survenu.

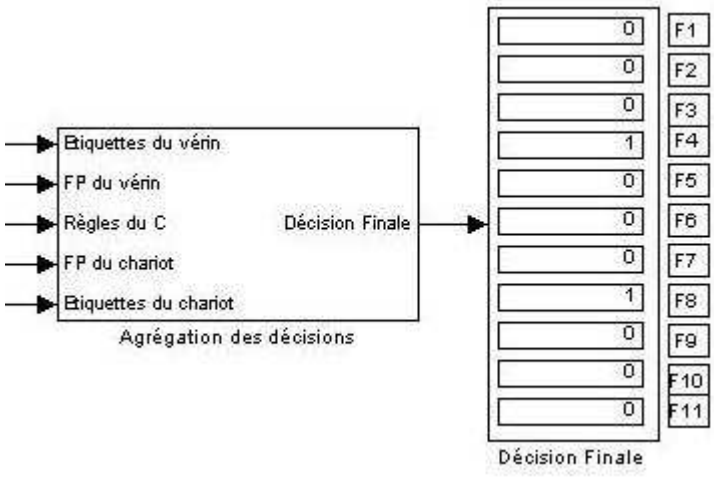


Figure 4-51 : Agrégation des décisions

4.4.5. Exemple de simulation de défaut

Une fois l'ensemble des diagnostiqueurs et le coordinateur implanté, il est possible de simuler les réactions de chacun par des signaux représentant un fonctionnement normal et

défaillant pendant une certaine période ou par des interrupteurs d'entrées/sorties.

Par exemple, lors d'une simulation nous avons obtenu les résultats de la Figure 4-52 où la décision finale a retourné l'étiquette F1 à l'utilisateur. L'étiquette F1 correspond à un défaut de la partition $\Pi_{F1} = \{f_1, f_2, f_3, f_4\}$ localisé sur le détecteur *a*. L'identification de ce défaut n'est pas toujours réalisable. Cependant, en regardant l'historique des événements présents dans chaque module des diagnostiqueurs locaux, il est possible de voir que le scénario survenu est le suivant : Activation de l'ordre droite de la commande entraînant le déplacement du chariot qui va désactiver le détecteur *a*. Cependant, l'historique du chariot nous montre qu'avant d'arriver au détecteur *b*, un événement externe a provoqué une activation du détecteur *a*. Le défaut est donc bien localisé sur le détecteur *a* avec une piste sur l'identification de l'origine du défaut par un événement extérieur déclencheur du détecteur.

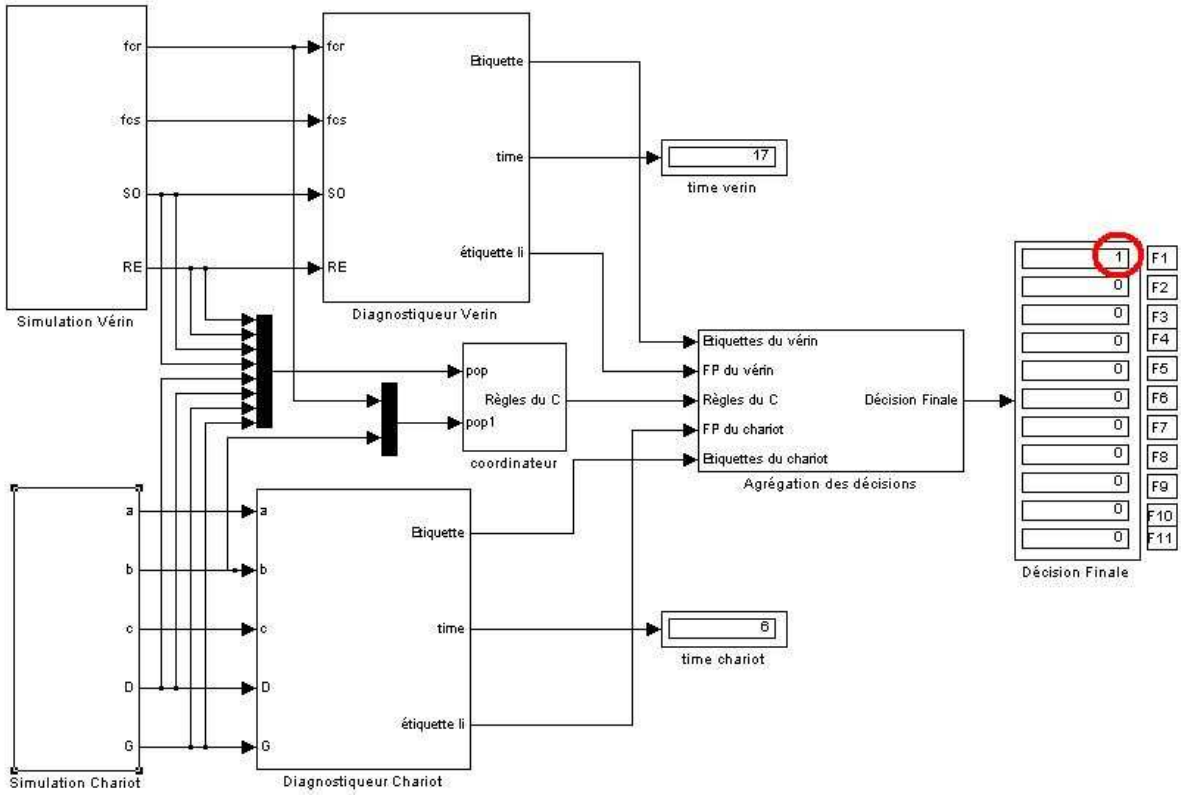


Figure 4-52 : Exemple de simulation

L'outil de simulation développé a l'avantage de pouvoir tester la réactivité notre approche de diagnostic décentralisée avec coordinateur pour différents types de défauts simulés. Il permet notamment de montrer que le diagnostic est rendu possible avec des performances équivalentes à une structure centralisée et cela malgré les différentes ambiguïtés ou cas d'indécision existants.

4.5. Conclusion

L'objectif principal d'une approche de diagnostic décentralisée dans le cadre des SED est de répondre au problème d'explosion combinatoire tout en garantissant les performances d'une approche centralisée. Ce travail demande une connaissance du procédé dans sa

technologie mais aussi dans son cahier des charges afin d'établir correctement le comportement désiré. Dans ce cadre de travail, nous avons proposé une approche de diagnostic décentralisée pour les applications manufacturières dans le chapitre 3. Nous avons vu que les systèmes manufacturiers peuvent être décomposés en éléments simples par leurs actionneurs associés à des détecteurs. Cette modularité a permis de constituer une bibliothèque d'Eléments de Partie Opérative (EPO). C'est cette information décentralisée des systèmes manufacturiers qui permet de réaliser une approche de diagnostic décentralisée. Cette approche demande à être expérimentée à travers différents exemples afin d'être validée et implantée sur des systèmes réels. Cette expérimentation, autour de deux exemples d'application manufacturière, a fait l'objet du chapitre 4.

Ce chapitre 4 a permis, dans un premier temps, de montrer l'intérêt de l'aspect décentralisé du diagnostic en terme d'explosion combinatoire. Nous avons donc montré à travers un exemple simpliste que l'apport du diagnostic décentralisé est considérable en termes d'états et de transitions tout en gardant les mêmes performances par la vérification de la capacité à diagnostiquer les partitions de défauts par la notion de co-diagnosticabilité mixte. Il paraît improbable de réaliser un diagnostic des SED centralisé pour un système industriel complexe.

Dans le cadre de cette étude sur l'explosion combinatoire, nous avons également comparé les deux algorithmes d'intersection entre la PO et la PC. Ces algorithmes font appel à des étapes de composition d'automates et de restriction du langage. Il en ressort que l'algorithme 2 d'intersection, privilégiant une restriction du langage de la commande avant sa composition synchrone avec l'Elément de Partie Opérative concerné, permet d'éviter un passage d'explosion d'états tout en gardant les mêmes performances que l'algorithme 1 d'intersection.

Le second exemple de ce chapitre est un système manufacturier plus complexe mis en place afin de montrer comment obtenir des diagnostiqueurs locaux et comment est traité un cas d'interactions entre des détecteurs communs à plusieurs diagnostiqueurs. En effet, les systèmes manufacturiers doivent, très souvent, partager une ressource et/ou un produit. Les approches de diagnostic décentralisées doivent répondre à ces interactions. Pour cela, le coordinateur, mis en place, doit intégrer des règles permettant de lever tous les cas d'indécision et ambiguïtés de décisions. Le coordinateur doit tout d'abord décrire l'ensemble des spécifications globales qui ne peuvent pas être établies par les diagnostiqueurs locaux. Ensuite, il doit regrouper l'ensemble des décisions locales des diagnostiqueurs et les décisions provenant des règles globales du coordinateur afin de retourner une décision finale à l'utilisateur qui soit cohérente et juste.

Pour finir ce chapitre, nous avons présenté un outil de simulation basé sur *Stateflow* de Matlab permettant de modéliser à la fois les diagnostiqueurs locaux et le coordinateur. Cet outil permet ainsi de valider notre approche par simulation du fonctionnement normal et anormal à travers différents scénarii. Au final, cette approche a permis de fournir un diagnostic réactif et performant vérifié par la notion de co-diagnosticabilité mixte.

Une fois que la structure décentralisée des diagnostiqueurs est mise au point et testée correctement, le code C obtenu par le compilateur peut, grâce au logiciel temps réel *Real Time Workshop*, être testé sur un Automate Programmable Industriel (API) en utilisant l'interface M2PLC. Cette interface génère une version exécutable du code C pour les API. Cependant, ce code ne peut pas être intégré dans tous les API. En fait, l'interface M2PLC cherche les entrées nécessaires au code C et va réaliser le traitement sur ces entrées, puis les résultats sont envoyés par M2PLC vers des registres spécifiques de l'API.

CONCLUSION GENERALE

Nous avons proposé une démarche globale de diagnostic décentralisé avec coordinateur pour détecter et localiser tous les défauts possibles pouvant survenir sur des procédés manufacturiers. Cette approche repose sur la construction de modèles enrichis au niveau du procédé, tenant compte de l'information apportée par la Partie Opérative (PO), de l'information issue de la commande et de l'information temporelle représentant les réactivités temporelles des actionneurs. Chacun de ces modèles représente ainsi le comportement local désiré de chaque élément de la partie opérative composé d'actionneurs et de capteurs discrets. La construction des diagnostiqueurs locaux repose alors sur ces modèles et permet de détecter et isoler les défauts qui entraînent la violation du comportement local désiré. Pour gérer la décision de l'ensemble des diagnostiqueurs locaux, la démarche propose de construire un coordinateur dont l'objectif est de garantir un diagnostic équivalent à celui d'un diagnostiqueur global. La construction du coordinateur ne nécessite pas d'avoir un modèle global du procédé, mais de prendre en compte un ensemble de règles permettant de lever d'éventuels conflits décisionnels ou de cas d'indécisions entre les diagnostiqueurs locaux. En effet, les diagnostiqueurs locaux n'observent que partiellement le procédé. Pour vérifier que la démarche proposée avec coordinateur diagnostique effectivement l'ensemble des défauts diagnosticables par un diagnostiqueur global, nous avons introduit formellement une notion de co-diagnosticabilité mixte reposant sur la prise en compte des événements, des états et de l'information temporelle. Cette notion garantit que chaque défaut, appartenant à l'ensemble des défauts, est diagnosticable par au moins un diagnostiqueur local ou par le coordinateur.

La construction des diagnostiqueurs ainsi que celle du coordinateur est réalisée hors ligne. L'implantation en ligne de la démarche s'effectue par l'intégration des diagnostiqueurs et du coordinateur dans un filtre intercalé entre la commande, implantée en ligne par l'utilisateur, et la partie opérative. Ainsi, tout ordre, ou tout compte-rendu d'exécution, est validé avant qu'il ne soit envoyé soit vers la partie opérative, soit vers la partie commande. L'avantage du concept de filtre est de limiter les conséquences souvent néfastes de la propagation des défauts provenant d'une erreur de commande ou d'un problème au niveau de la partie opérative (actionneurs ou/et capteurs).

Les travaux que nous avons présentés dans ce mémoire s'inscrivent dans une continuité logique de recherche de l'équipe Automatique – SED au sein du laboratoire CReSTIC. En effet, après des travaux sur la vérification et la synthèse de commande [NDJ 99], [PHI 03] et [TAJ 05], cette thèse propose d'utiliser certains résultats obtenus pour renforcer à la fois l'aspect modélisation de la partie opérative et surtout pour réaliser une méthode décentralisée pour le diagnostic des défaillances des Systèmes à Evénements Discrets. Plusieurs perspectives peuvent être envisagées comme suite du travail de cette thèse et s'envisager à court terme ou à long terme. Nous les avons regroupées selon quatre thèmes : applicabilité de la démarche sur des systèmes réels, extension de la démarche à la prise en compte de nouvelles fonctionnalités, évolution de la démarche vers une approche non plus décentralisée mais distribuée et intégration de la démarche dans une architecture de commande :

Applicabilité de la démarche sur des systèmes réels

L'applicabilité de l'approche consiste, dans un premier temps, à réaliser l'implantation de l'outil de simulation dans un environnement réel d'applications industrielles afin d'effectuer le diagnostic en ligne. Les Automates Programmables Industriels (API) sont largement utilisés dans le milieu industriel pour l'automatisation des installations industrielles. Deux techniques sont possibles. La première solution est d'effectuer l'implantation de la démarche directement dans un Automate Programmable Industriel à travers un langage de programmation. Pour cela, nous proposons d'allouer une zone mémoire de l'API, aux données nécessaires pour les calculs et les résultats de l'approche de diagnostic. Tous les ordres envoyés par la PC et les comptes-rendus de la PO sont d'abord enregistrés dans des adresses spécifiques de la zone mémoire avant d'être validés par le module de diagnostic. Une seconde solution est d'utiliser l'outil de simulation développé sous Matlab avec l'API. En fait, tout code Matlab de version 7.0 peut être considéré comme client OPC. Une connexion à un serveur OPC à partir de Matlab permet alors d'accéder aux variables de l'API et d'effectuer une lecture/écriture sur les différentes variables. Ainsi, les informations disponibles sur la commande et/ou sur la PO peuvent être envoyées au module de diagnostic codé par *Stateflow* de Matlab. Les informations sur la décision de diagnostic, calculées par cet outil, peuvent être récupérées par l'API et mémorisées dans un registre spécifique permettant ainsi de prendre une décision soit sur la validation de l'ordre ou du compte rendu, soit sur la déclaration d'un défaut.

Pour rendre la démarche réellement utilisable par un concepteur quelconque, l'idée est de développer une Interface Homme Machine permettant, d'une part, d'aider un Opérateur Humain de Supervision à mieux comprendre l'état fonctionnel du procédé et donc de suivre, d'une façon efficace, son fonctionnement et d'autre part, de faciliter l'acquisition des connaissances de l'expert en lui offrant la possibilité d'intégrer de nouvelles règles ou de nouvelles contraintes, et de tester leurs effets sur le diagnostic et le suivi du procédé.

Extensions possibles de l'approche

Dans la proposition que nous avons faite dans ce travail de thèse, nous avons modélisé le procédé manufacturier d'un point de vue équipement, c'est-à-dire du point de vue des actionneurs et des capteurs. Cependant, le problème des événements concurrentiels, liés au cas de déplacement ou de traitement de plusieurs produits, n'a pas été pris en compte. Une extension possible consiste donc à développer la démarche de diagnostic vers la prise en compte des événements concurrentiels. Pour cela, il faut effectuer un modèle du produit en plus des modèles de PO. De plus, un système de production gère très souvent un ensemble de produits de différents types où chaque type de produits répond à des contraintes particulières. Une solution, pour réaliser un modèle de produits, est d'utiliser les Réseaux de Petri colorés (RdPs) comme outil de modélisation parce que les jetons de couleur peuvent être associés à des types spécifiques de produits.

Une deuxième extension concerne l'utilisation de la logique floue pour la construction des fonctions de prévision. En effet, celle-ci semble efficace puisqu'elle fournit un indicateur graduel de la violation d'une contrainte temporelle. L'évolution de la valeur fournie par la fonction de prévision d'un composant (actionneur par exemple) de "0", qui correspond au respect total d'une contrainte temporelle, vers la valeur "1", correspondant à la violation sûre de la contrainte, permet de prévoir l'évolution du fonctionnement du composant vers un défaut. Dès lors, il est envisageable de développer la démarche pour réaliser une fonction de pronostic permettant de prévoir l'occurrence de tous les défauts de nature progressive. De

plus, notre approche permet de détecter les défauts de la PO mais également les erreurs de programmation de la PC implantée. Dès lors, notre approche de diagnostic peut être utilisée pour la vérification de la commande implantée.

L'information disponible sur les Systèmes à Evénements Discrets est très limitée, une troisième extension possible est d'intégrer la notion de capteur virtuel pour augmenter la quantité d'informations disponibles sur le fonctionnement du système complexe qui possède un nombre limité de capteurs. Cette information peut être obtenue grâce au traitement statistique des signaux issus des capteurs réels.

Orientation de l'approche vers une structure distribuée

Il faut, pour réaliser une structure distribuée de diagnostic, développer un protocole de communication entre les diagnostiqueurs locaux. Pour chaque diagnostiqueur local, un ensemble de règles va définir les événements qui doivent être communiqués et à qui, afin que les diagnostiqueurs locaux résolvent tous les conflits décisionnels ou tous les cas possibles d'indécision. Ces règles devront également prendre en compte les délais de communication. Il faut par conséquent, développer une méthodologie permettant, d'une part, de définir, pour chaque diagnostiqueur local, l'ensemble des événements nécessaires devant être communiqués aux autres diagnostiqueurs et d'autre part, de vérifier la cohérence de toutes les décisions locales résultantes de l'application des règles de communication.

Intégration de la démarche en vue d'une architecture de commande

Une perspective à plus long terme de cette démarche, est d'établir une procédure de reconfiguration de la commande après défaillance. Ce travail demande à la fois une classification des défauts selon leur importance et une identification précise du défaut. La démarche de développement d'une telle reconfiguration de la commande repose sur l'analyse préliminaire des risques, une définition des situations critiques, l'intégration des fonctions de surveillance et de reprise, et la mise en oeuvre de l'architecture industrielle de commande [NIE 98], [GHA 04]. Une solution, pour la reconfiguration de la commande à partir de notre approche, est d'analyser les étiquettes de défauts et d'indiquer la situation du système à l'utilisateur. L'utilisateur doit avoir la possibilité de choisir le séquencement d'événements lui permettant de retourner vers un fonctionnement sans risque.

BIBLIOGRAPHIE

[ALL 98] Alla H., David R., "Continuous and hybrid Petri nets", Journal of Circuits, Systems and Computer, 8 (1), pp.159-188, 1998.

[ANI 00] Anil K.J., ROBERT P.W., MAO J., "Statistical Pattern Recognition: A review", IEEE Transactions on pattern analysis and machine intelligence 1 (22), pp.4-36, 2000.

[ARN 92] Arnold A., "Systèmes de transitions finis et sémantique des processus communicants", Masson, 1992.

[BAC 92] Baccelli F., Cohen G., Olsder G.J., Quadrat J.P., "Synchronization and linearity. An algebra for Discrete Event Systems", Wiley, 1992.

[BAL 93] Balemi S., Hoffmann G.J., Gyugyi P., Wong-Toi H., Franklin G.F., "Supervisory control of a rapid thermal multiprocessor", Proceeding IEEE Transactions on Automatic Control, Vol. 38, N°7, pp.1040-1059, 1993.

[BEN 03] Benveniste A., Haar S., Fabre E., Jard C., "Distributed monitoring of concurrent and asynchronous systems", In Proceeding ATPN-Workshop on Discrete Event Systems Control, Eindhoven, The Netherlands, pp.97–142, June 2003.

[BOE 04] Boel B.K., Jiroveanu G., "Distributed contextual diagnosis for very marge systems", 7th International Workshop on Discrete Event Systems WODES'04, pp.343-348, Reims, France, 2004.

[BOU 03a] Boufaïed A., "Contribution à la Surveillance Distribuée Des Systèmes à Evénements Discrets Complexes", Thèse de l'université Paul Sabatier, Toulouse, 2003.

[BOU 03b] Boufaïed A., Subias A., Combacau M., "Détection Distribuée : Reconnaissance Floue de Chroniques Distribuées", Rapport LAAS 03300, octobre 2003.

[BOU 04] Boufaïed A., Subias A., Combacau M., "Distributed Fault Detection with Delays Considerations", Rapport LAAS 04099, février 2004.

[BOU 98] Bousson K., Steyer J., Travé-Massuyès L., Dahhou B., "From a rule-base to a predictive qualitative model-based approach using automated model generation. Application to the monitoring and diagnosis of biological process", Engineering Applications of Artificial Intelligence, 11, pp.477-493, 1998.

[BOU 96] Boutleux E., "Diagnostic et suivi d'évolution de l'état d'un système par reconnaissance des formes floues. Application au modèle du réseau téléphonique français", Thèse présentée devant l'Université de Technologie de Compiègne, 1996.

[BRA 83] Brams G.W., "Réseaux de Petri : théorie et pratique", Masson, 1983.

[BRU 90] Brunet J., Labarrere M., Jaume D., Rault A., Verge M., “Détection et diagnostic de pannes : approche par modélisation”, Traité des Nouvelles Technologies, série Diagnostic et Maintenance, Hermès, 1990.

[CAL 90] Calvez J.P., “Spécification et conception des systèmes”, Masson, 1990.

[CAR 02] Carré-Ménétrier V., Zaytoon J., “Grafcet: behavioural issues and control synthesis”, European Journal of Control (EJC), Vol. 8, N°4, pp.375-401, 2002.

[CAR 03] Carré-Ménétrier V., “Synthèse formelle de la commande opérationnelle des SED”, Présentation aux Journées Nationales d’Automatique JNA’03, Valenciennes, pp.31-32, 25-27 juin 2003.

[CAS 99] Cassandra C.G., Lafortune S., “Introduction to Discrete Event Systems”, Kluwer Academic Publisher, ISBN 0792386094, 1999.

[CHA 93] Chand S., “Discrete-event based monitoring and diagnosis of manufacturing processes”, Proceeding of the American control conference, San Francisco, California, pp.1508-1512, June 1993.

[CHA 01] Chandra V., Kumar R., “A Discrete Event Systems Modeling Formalism Based on Event Occurrence Rules and Precedences”, In IEEE Transactions on Robotics and Automation, Vol. 17, N°6, 2001.

[CHA 99] Charbonnier F., Alla H., David R., “The Supervised Control of Discrete-Event Dynamic Systems”, In IEEE Transactions on Control Systems Technology, Vol. 7, N°2, 1999.

[COH 95] Cohen G., “Cours : Théorie algébrique des systèmes à événements discrets”, Centre Automatique des Systèmes, Ecole des Mines de Paris, Fontainebleau et INRIA Rocquencourt, 1995.

[COM 91] Combacau M., “Commande et surveillance des systèmes à événements discrets complexes : application aux ateliers flexibles”, Thèse, Toulouse, 1991.

[COM 98] Combacau M., Zamai E., Subias A.,“A discrete events model of a monitoring strategy as a control structure of a monitoring system”, Proceeding of the IMACS CESA’98 conference, Hammamet, Tunisie, Avril 1998.

[COM 00a] Combacau M., Berruet P., Charbonnaud F., Khatab A., Zamai E., “Supervision and monitoring of production systems”, Proceeding of MCPL'2000, Grenoble, 4-6 juillet 2000.

[COM 00b] Combacau M., Berruet P., Charbonnaud F., Khatab A., “Réflexions sur la terminologie : Surveillance – Supervision”, Groupement pour la Recherche en Productique, Systèmes de Production Sûrs de Fonctionnement, http://www.laas.fr/~combacau/SPSF/sursup.html, 03 mars 2000.

[COM 01] Combacau M., “Approche discrète de la surveillance des systèmes de production”, Traité IC2 Hermès, Maîtrise des risques et sûreté de fonctionnement des systèmes de production, Chapitre 11, Septembre 2001.

[CON 04] Contant O., Lafortune S., Teneketzis D., “Diagnosis of modular discrete event systems”, 7th International Workshop on Discrete Event Systems WODES’04, pp.337-342, Reims, France, 2004.

[CRU 91] Cruette D., “Méthodologie de conception des systèmes complexes à événements discrets : application à la conception et à la validation hiérarchisée de la commande de cellules flexibles de production dans l’industrie manufacturière”, Thèse, Lille, 1991.

[DAN 97] Dangoumau N., “Commande et Surveillance des Systèmes à Evénements Discrets, Approche par Réseaux de Petri”, DEA Automatique et Informatique Appliquée, Ecole centrale de Nantes, 1997.

[DAR 03] Darkhovski B., Staroswiecki M., “Theoretic Approach to Decision in FDI”, IEEE Transactions On Automatic Control, Vol. 48, N°5, 2003.

[DAS 03a] Da Silveira M., “Sur la Distribution avec Redondance Partielle de Modèles à Evénements Discrets Pour la Supervision de Procédés Industriels”, Thèse de l’université Paul Sabatier, Toulouse, 2003.

[DAS 03b] Da Silveira M., Combacau M., “Distribution de Modèles à Evénements Discrets : Procédure systématique basée sur les Réseaux Petri”, Rapport LAAS 03057, Octobre 2003.

[DAV 89] David R., Alla H., “Du Grafcet au réseaux de Petri”. Hermès, 1989.

[DEB 00] Debouk R., Lafortune S., Teneketzis D., “Coordinated decentralized protocols for failure diagnosis of discrete event systems”, Discrete Event Dynamic Systems: Theory and Applications, Vol. 10, N°1-2, pp.33-86, January 2000.

[DEV 00] Devaux S., Rachline M., “Introduction à l’automatisme: Schneider Electric”, Encyclopédie économie 3000, série haute technologie. ISBN 2-7191-0551-1, 2000.

[DUB 90] Dubuisson B., “Diagnostic et reconnaissance des formes”, Traité des Nouvelles Technologies, série Diagnostic et Maintenance, Hermès, 1990.

[DUB 01] Dubuisson B., “Automatique et statistiques pour le diagnostic”, Traité IC2 Information, commande, communication. Hermès Sciences, 2001.

[DUD 01] Duda R.O., Hart P.E., Stork D.E., “Pattern Classification”, Wiley, New York, second edition, 2001.

[FAB 02] Fabre E., Benveniste A., Jard C., “Distributed diagnosis for large discrete event dynamic systems”, In Proceeding 15th IFAC World Congress, Barcelona, Spain, July 2002.

[FER 04] Ferrier J.L., Boimond J.L., “Modèles et Systèmes. Chapitre 2 : Modèles pour les Systèmes à Evénements Discrets”, cours de DEA, Mise à niveau et Tronc commun, http://www.istia.univ-angers.fr/~ferrier/#ancre4, Juin 2004.

[FRA 90] Frank P., “Fault diagnosis in dynamic systems using analytical and knowledge based redundancy - a survey and some new results”, Automatica, Vol. 26, pp.459-474, 1990.

[GAR 05] Garcia H.E., Yoo T.S., “Model-based detection of routing events in discrete flow networks”, Automatica, Vol. 41, 2005.

[GEN 03] Genc S., Lafortune S., “Distributed diagnosis of discrete-event systems using Petri nets”, Proceeding 2003 International Conference on Application and Theory of Petri Nets, pp.316-336, Eindhoven, The Netherlands, June 2003.

[GEN 00] Gendreau D., Bodart A., Carré-Ménétrier V., De Loor P., Deluche J.B., Dupont J., Hancq J., Kril A., Nido J., “7 facettes du GRAFCET : Approches pratiques de la conception à l’exploitation”, Automatisation & Production, ISBN 2.85428.462.5, 2000.

[GHA 96] Ghallab M., “On chronicles: representation, on-line recognition and learning”, 5th International Conference on Principles of Knowledge Representation and Reasoning KR’96, Cambridge, USA, pp.597-606, 1996.

[GHA 04] Ghariani A., Toguyeni A., Craye E., “Towards an approach to automate reconfiguration procedure in Automated Production Systems”, In IEEE SMC 2004, International Conference on Systems, Man and Cybernetics, The Hague, The Netherlands, October 10-13 2004.

[GOU 03] Gouyon D., Pétin J.F., Gouin A., “Modèles du procédé et de ses spécifications pour la synthèse de la commande”, Colloque Francophone sur la modélisation des systèmes réactifs, MSR’03, Metz, France, pp.45-60, 6-8 octobre 2003.

[HAR 87] Harel D., “Statecharts: a visual formalism for complex systems”, Science of Computer Programming, Vol. 8, N°6, pp.231-274, 1987.

[HO 89] Ho Y.C., “Special Issue on Dynamics of Discrete Event Systems” Proceeding IEEE, Vol. 77, N°1, janvier 1989.

[HOL 90] Holloway L.E., Krogh B.H., “Fault detection and diagnosis in manufacturing systems”, International Conference on Computer Integrated Manufacturing, Vol. 2, pp.252-259, 1990.

[HOL 91] Holloway L.E., Krogh B.H., “Monitoring behavioral evolution for on-line fault detection”, Proceeding of the IFAC SAFEPROCESS’91 conference, Baden Baden, 1991.

[HOL 94] Holloway L.E., Chand S., “Time templates for discrete event fault monitoring in manufacturing systems”, In American Control Conference, Baltimore, MD, 1994.

[IEC 93] International Electrotechnical Commission, “PLCs – Part 3: programming languages”, Publication 611131-3, 1993.

[IEC 02] International Electrotechnical Commission, “GRAFCET specification language for sequential function charts”, IEC 60848, 2002.

[IGL 89] IGL Technology, “SADT, un langage pour communiquer”, Editions EYROLLES, 1989.

[KAT 04] Kattan B., “Synthèse structurelle d’un contrôleur basée sur le Grafcet”, Thèse de l’université Joseph Fourier – Grenoble 1, 2004.

[KHA 93] El Khattabi S., “Intégration de la surveillance de bas niveau dans la conception des systèmes à événements discret”, Thèse de Doctorat, Lille, France, 1993.

[KLE 05] Klein S., Litz L., Lesage J.J., “Fault Detection of Discrete Event Systems Using an Identification Approach”, In 16th IFAC World Congress, CDRom paper n°02643, 6 pages, Praha (CZ), July 4-8, 2005.

[KOU 98] Koutsoukos X., Lemmon M., Antsaklis P., “Timed Petri nets in Hybrid Systems: Stability and Supervisory Control, Journal of Discrete Event Dynamic Systems: Theory and Applications”, 8(2), pp.137-173, 1998.

[KOU 00] Koutsoukos X., Antsaklis P., Stiver J., Lemmon M., “Supervisory Control of Hybrid Systems”, Proceeding of IEEE, Special Issue in Hybrid Systems, pp.1026-1049, 2000.

[KUM 91] Kumar R., “Supervisory Synthesis Techniques for Discrete Event Dynamical Systems”, Thesis for Ph. D. Degree, University of Texas, 1991.

[LHO 91] Lhoste P., “Surveillance des M.S.A.P.: les Atouts de la modélisation de Comportement”, Journée Surveillance du pôle SED (GT2) du GR Automatique, Paris, 1991.

[LHO 97] Lhoste P., Faure J.M., Lesage J.J., Zaytoon J., “Comportement temporel du Grafcet”, Journal Européen des Systèmes Automatisé (JESA), Vol. 31, N°4, Hermès, pp.695-711, 1997.

[LIN 94] Lin F., “Diagnosability of Discrete Event Systems and its Applications”, In Discrete Event Dynamic Systems, 4, Kluwer Academic Publishers, Boston, USA, pp.197-212, 1994.

[MAN 00] Manders E.J., Biswas G., Narasimah S., Mosterman P.J., “Combined Qualitative, Quantitative approach for fault isolation in continuous dynamic systems”, In 4th Symposium on Fault Detection Supervision and Safety of Technical Processes, Budapest, Hungary, 2000.

[MAT 04] The MathWorks, “Stateflow 6: Design and simulate event-driven systems”, www.mathworks.com, 2004.

[MON 00] Montmain J., Gentil S., “Dynamic causal model diagnostic reasoning for online technical process supervision”, Automatica, 36, pp.1137-1152, 2000.

[MOS 01] Mosterman J., “Diagnosis of Physical Systems With Hybrid Models Using Parameterised Causality”, Proceeding of Hybrid Systems: Computation and Control, 4th International Workshop (HSCC01), Rome, Italia, pp.447-458, 2001.

[MOU 02] Sayed Mouchaweh M., “Conception d’un système de diagnostic adaptatif et prédictif basé sur la méthode Fuzzy Pattern Matching pour la surveillance en ligne des systèmes évolutifs : Application à la supervision et au diagnostic d’une ligne de peinture au trempé”, Thèse soutenue à l’Université de Reims, 11 Décembre 2002.

[MOU 04] Sayed Mouchaweh M., Philippot A., Triki S., Riera B., “Separated approach for active monitoring of Discrete Event Systems”, In 7th International Workshop On Discrete Event Systems WODES’04, Reims, France, pp.391-396, September 2004.

[MOU 05] Sayed Mouchaweh M., Philippot A., Carré-Ménétrier V., Riera B., "Detectability and Diagnosability of Discrete Event Systems", 2nd International Conference on Informatics in Control, Automation and Robotics, ICINCO'05, Barcelona, Spain, 14-17 september 2005.

[MOU 06] Sayed Mouchaweh M., Philippot A., Carré-Ménétrier V., Riera B. "Timed-Event-State-Based Diagnoser for Manufacturing Systems", 7th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services, BASYS'06, Niagara Falls, Ontario, Canada, 4-6 september 2006.

[MUR 89] Murata T., "Petri nets : properties, analysis and applications", Proceeding IEEE, Vol. 77, N°4, pp.541-580, avril 1989.

[NDJ 99] Ndjab C.H., "Synthèse de la commande des SED par GRAFCET", Thèse de l'Université de Reims Champagne Ardenne, 1999.

[NIE 98] Niel E., Rezg N., "Reconfiguration de la commande et sécurité opérationnelle : application", Techniques de l'Ingénieur, dossier R7641, Vol. S2, mars 1998.

[PAN 00] Pandalai D., Holloway L.E.N., "Template Languages for Fault Monitoring of Timed Discrete Event Processes", In IEEE Transactions On Automatic Control, Vol.45, N° 5, 2000.

[PER 04] Perrin J., Binet F., Dumery J.J., Merlaud C., Trichard J.P., "Automatique et informatique industrielle : Bases théoriques, méthodologiques et techniques", Nathan Technique, ISBN 2-09-179452-X, novembre 2004.

[PER 84] Perrow C., "Normal Accidents: Living with High Risk Technologies", Basic Books, Inc., New York, 1984.

[PET 81] Peterson J.L., "Petri Net Theory and the modelling of systems" Prentice Hall, Englewood Cliffs, 1981.

[PHI 02] Philippot A., "Implantation d'une commande sûre dans un Automate Programmable Industriel à partir d'une spécification exprimée en Grafcet", Mémoire de DEA TORIC en Optimisation et Sûreté des Systèmes, Reims, Septembre 2002.

[PHI 03] Philippot A., Tajer A., Gellot F., Carré-Ménétrier V., "Synthèse de la commande spécifiée en Grafcet: application à un préhenseur pneumatique", Colloque Francophone sur la modélisation des systèmes réactifs, MSR'03, Metz, France, pp.61-75, 6-8 octobre 2003.

[PHI 04a] Philippot A., Tajer A., Gellot F., Carré-Ménétrier V., "On-line synthesis approach based on a structured plant modelling", In 7th International Workshop On Discrete Event Systems WODES'04, Reims, France, pp.397-402, September 2004.

[PHI 04b] Philippot A., Tajer A., Gellot F., Carré-Ménétrier V., "Méthodologie de modélisation dans le cadre de la synthèse formelle des SED", In Conférence Internationale Francophone d'Automatique (CIFA'04), Douz, Tunisie, 2004.

[PHI 04c] Philippot A., Tajer A., Gellot F., Carré-Ménétrier V., "On-line synthesis approach based on a structured plant modelling", Numéro spécial e-STA sur les SED, revue en ligne de la SEE, Vol. 1, n°3, 2004.

[PHI 05a] Philippot A., Sayed Mouchaweh M., Carré-Ménétrier V., "Multi-models approach for the diagnosis of Discrete Events Systems", In International conference on Modelling, Analyse and Control of Dynamic Systems (IMACS'05), Paris-France, 2005.

[PHI 05b] Philippot A., Sayed Mouchaweh M., Carré-Ménétrier V., "Diagnostic des SED par modélisation multi-outils", JDMACS'05, Lyon, France, 5-7 septembre 2005.

[PHI 05c] Philippot A., Tajer A., Gellot F., Carré-Ménétrier V., "Méthodologie de modélisation dans le cadre de la synthèse formelle des SED", Numéro spécial e-STA sur les SED, revue en ligne de la SEE, Vol. 2, n°2, 2005.

[PHI 05d] Philippot A., "Une approche multi-outils décentralisée et modulaire pour le diagnostic des Systèmes à Evénements Discrets", Présentation aux Journées du Pôle STP du GDR MACS, Clermont Ferrand., 31 mars-1er avril 2005.

[PHI 06] Philippot A., Sayed Mouchaweh M., Carré-Ménétrier V., Riera B., "Decentralized Approach to Diagnose Manufacturing Systems", In Computational Engineering in Systems Applications CESA'06, Beijing, China, 2006.

[QIU 04] Qiu W., Kumar R., "Decentralized failure diagnosis of discrete event systems" 7th International Workshop on Discrete Event Systems WODES'04, pp.145-150, Reims, France, 2004.

[QIU 05] Qiu W., "Decentralized / distributed failure diagnosis and supervisory control of discrete event systems", Thesis, Iowa State University, 2005.

[RAM 89a] Ramadge P.J.G., Wonham W., "Supervisory control of a class of discrete events systems", SIAM journal of Control and Optimization, Vol. 25, N°1, 1989.

[RAM 89b] Ramadge P.J.G., Wonham M., "The Control of Discrete Event Systems", Proceeding IEEE, Vol. 77, N°1, January 1989.

[RAY 04] Rayhane H., "Surveillance des systèmes de production automatisées : Détection et Diagnostic", Thèse en cotutelle à l'INP de Grenoble, 2004.

[RIC 01] Ricker S.L., Schuppen J.H.V., "Decentralized failure diagnosis with asynchronous communication between supervisors", In Proceeding European Control Conference, pp.1002-1006, 2001.

[RIE 04] Riera B., Gueguen H., Moreno S., Debernard S., Bonte T., "La supervision : outils et applications". Revue de l'Electricité et de l'Electronique, REE, N°4, pp.27-51, Avril 2004.

[ROH 06] Rohée B., Riera B., Carré-Ménétrier V., Roussel J.M., "A methodology to design and check a plant model", 3rd IFAC Workshop on Discrete Event System Design DESDes'06, Leszno and Rydzyna, Poland, 26-28 September 2006.

[ROU 04] Rouelle K., "Stateflow : La modélisation de systèmes événementiels dans l'environnement Matlab/Simulink", The MathWorks Matlab&Simulink release 14, 2004.

[ROU 94] Roussel J.M., "Analyse de Grafcets par Génération Logique de l'Automate Equivalent", Thèse de l'Ecole Normale Supérieure de Cachan, 1994.

[ROU 03] Roussel J.M., Medina A., Faure J.M., “Synthèse d’un programme de commande d’un système logique à partir de l’expression algébrique de ses spécifications”, Colloque Francophone sur la modélisation des systèmes réactifs, MSR’03, Metz, France, pp.77-93, 6-8 octobre 2003.

[ROU 05] Roussel J.M., Giua A., “Designing Dependable Logic Controllers Using the Supervisory Control Theory”, In 16th IFAC World Congress, CDRom paper n°04427, 6 pages, Praha (CZ), July 4-8, 2005.

[SAH 87] Saharaoui A.E.K.,“Vers une approche de Surveillance des systèmes”, Thèse Doctorat Université Paul Sabatier, Toulouse, Octobre 1987.

[SAM 94] Sampath M., Sungupta R., Lafortune S., Sinnamohideen K., Teneketzis D., “Diagnosability of discrete event systems”, In 11th International Conference Analysis Optimization of Systems: Discrete Event Systems, Sophia-Antipolis, France,1994.

[SAM 95] Sampath M., “A Discrete Event Systems Approach to Failure Diagnosis”, Thesis, University of Michigan, 1995.

[SAM 01] Sampath M., “A hybrid approach to failure diagnosis of industrial systems”, In Proceeding American Control Conference - ACC’01, Arlington, VA, USA, June 2001.

[SCH 87] Scherer W.T., White C.C., “A survey of expert systems for equipment maintenance and diagnostics”, In Fault Detection and Reliability: Knowledge Based & Other Approaches, M. G. Singh, K. S. Hindi, G. Schmidt, and S. Tzafestas, editors, Pergamon Press, 1987.

[SRI 93] Srinivasan V. S., Jafari M. A., “Fault detection/monitoring using time Petri nets”, Proceeding IEEE Transactions on Systems, Man and Cybernetics, Vol. 23, July/August 1993.

[STE 91] Steyer J.P., “Sur une approche qualitative des systèmes physiques: aide en temps réel à la conduite des procédés fermentaires”, Thèse de l’université Paul Sabatier, Toulouse, Décembre 1991.

[SU 03] Su R., Abdelwahed S., Karsai G., G. Biswas, “Discrete Abstraction and Supervisory Control of Switching Systems”, IEEE International Conference on Systems, Man and Cybernetics, Vol. 1, pp.415-421, 2003.

[SU 04] Su R., “Distributed Diagnosis for Discrete-Event System”, Thesis of PhD, University of Toronto, Canada, 2004.

[TAJ 03] Tajer A., Philippot A., Gellot F., Carré-Ménétrier V., “Contribution à l’amélioration de la praticabilité des approches formelles de synthèse de commande”, Journées Doctorales d’Automatique 2003, Valenciennes, pp.239-244, 25-27 juin 2003.

[TAJ 04] Tajer A., Philippot A., Gellot F., Carré-Ménétrier V., “Démarche formelle de synthèse de synthèse d’une commande sûre à partir d’une spécification Grafcet”, In Conférence Internationale Francophone d'Automatique (CIFA’04), Douz, Tunisie, 2004.

[TAJ 05] Tajer A., “Contribution aux approches formelles de synthèse de commande spécifiée par GRAFCET”, Thèse de l’Université de Reims Champagne Ardenne, 2005.

[TOG 92] Toguyeni A.K.A., “Surveillance et diagnostic en ligne dans les systèmes flexibles de l’industrie manufacturière”, Thèse de l’université de Lille 1, Lille, 1992.

[TOU 05] Touaf S., “Diagnostic logique des systèmes complexes dynamiques dans un contexte multi-agent”, Thèse Université Joseph Fourier, Grenoble 1, 2005.

[TRA 97] Travé-Massuyès L., Dague P., Guerrin F., “Le raisonnement qualitatif”, Hermès, France, 1997.

[TRI 04] Triki S., Sayed-Mouchaweh M., Riera B., Carre-Ménétrier V., Zaytoon J., “Human adapted qualitative diagnosis for abrupt faults”, IFAC 11th Symposium on Information Control Problems in Manufacturing (INCOM’O4), Salvador-Brazil, 2004.

[TRI 02] Tripakis S., “Fault Diagnosis for Timed Automata”, In Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT’02), Vol.2469 of *LNCS*, Springer, 2002.

[UNG 93] Ungauer C., “Problématique d’utilisation de techniques de supervision à base de connaissances profondes : l’exemple de la supervision du réseau TRANSPAC”, Technical report, France Telecom R&D, 1993.

[VAL 89] Valette R., Cardoso J., Dubois D., “Monitoring manufacturing systems by means of Petri nets with imprecise markings,” In Proceeding IEEE Conference Intelligent Control, Albany, NY, pp.233-238, Sept. 1989.

[WAN 00] Wang Y., “Supervisory Control of Boolean Discrete-Event Systems”, Thesis of Master of Applied Sciences, University of Toronto, Canada, 2000.

[WAN 04] Wang Y., Yoo T.S., Lafortune S., “New Results on Decentralized Diagnosis of Discrete Event Systems”, Annual Allerton Conference on Communication, Control and Computing, 2004.

[WON 95] Wonham W.M., “Notes on control of discrete event systems”, ECE 1636F/1637S, University of Toronto, Canada, 1995.

[ZAD 98] Zad S.H., Kwong R.H., Wonham W.M., “Fault diagnosis in discrete-event systems”, In CDC’98, IEEE Conference on Decision and Control, Tampa, Florida, USA, 1998.

[ZAD 03] Zad S. H., Kwong R. H., Wonham W. M., “Fault Diagnosis in Discrete Event Systems: Framework and model reduction”, Proceeding IEEE Transactions On Automatic Control, Vol. 48, N°7, 2003.

[ZAM 98] Zamai E. “Models and strategies for monitoring of flexibles manufacturing systems”, Proceeding of the IFAC INCOM’98 conference, Nancy, juin 1998.

[ZAY 01] Zaytoon J., “Systèmes dynamiques hybrides, sous la direction de J. Zaytoon”, traité IC2, Hermès-Paris, 378 pages, 2001.

[ZWI 95] Zwingelstein G., “Diagnostic des défaillances”, Traité des Nouvelles Technologies, série Diagnostic et Maintenance, Hermès, 1995.

Annexe 1

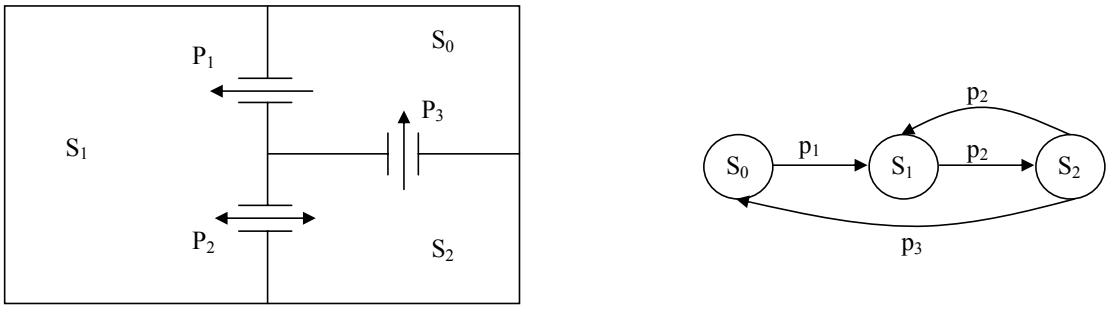
Systèmes à Evénements Discrets (SED)

La vitesse, l'accélération, le niveau, la pression, la température, le débit, la tension, le courant sont des variables continues, dans le sens où elles peuvent prendre n'importe quelle valeur lorsque le temps évolue. Les équations différentielles sont l'outil de base approprié pour la modélisation, l'analyse et la commande de ces systèmes [FER 04]. Les grandeurs discrètes représentent quant à elle, un nombre de produits dans un stock, un nombre de processeurs en activité, ou bien encore la position d'un produit. L'évolution de ces grandeurs discrètes est conditionnée par l'occurrence d'événements tels que la fin d'exécution d'une tâche, le franchissement d'un seuil, l'arrivée d'un produit, d'un client, la défaillance d'un dispositif [WON 95]. [FER 04] décrit un Système à Evénements Discrets (SED) comme « un système à espace d'états discret dont les transitions entre états sont associées à l'occurrence d'événements discrets asynchrones ». La succession d'événements constitue des trajectoires permettant de décrire cet espace d'états. Pour les modèles à événements discrets, l'espace d'états est un ensemble discret et l'état change seulement à certains instants du temps, de façon instantanée. Un état représente une situation du procédé. Le passage d'un état à un autre se caractérise par l'occurrence d'un événement appelé « transition » [HO 89], [COH 95] et [CAS 99].

Alors que les systèmes continus s'intéressent à des processus obéissant à des lois physiques, des équations différentielles ou aux dérivées partielles, les SED recouvrent des systèmes ne se souciant que des débuts et des fins des phénomènes (événements discrets) et à leur enchaînement dynamique, logique ou temporel [ZAY 01]. Une des manières de pouvoir classifier les SED est de définir le type d'application et le cadre de travail [COH 95] :

- Systèmes manufacturiers : les méthodes de production manufacturière modernes se sont tournées vers des ateliers flexibles organisés en réseaux de machines multi-tâches. La production simultanée de produits en série a conduit les systèmes à une plus grande souplesse d'utilisation mais aussi à une gestion plus difficile des équipements. Les systèmes manufacturiers possèdent surtout un aspect modulaire sur les composants rendant l'information disponible décentralisée.
- Systèmes informatiques : ils regroupent tous les systèmes constituant l'informatique en règle général en partant de la puce électronique au réseau d'ordinateurs.
- Réseaux de communication, EDF et de transport : ce sont les réseaux qui doivent gérer des situations de conflits tels que le téléphone ou le réseau ferroviaire où des messages, respectivement des trains d'informations, doivent circuler à travers une même voie sans se télescoper.
- Planification de tâches : la gestion d'un chantier de construction ou d'un projet nécessite la coordination de différentes tâches dont certaines peuvent se dérouler en parallèle sans ordre préétabli, alors que d'autres nécessitent l'apport de contraintes de précédence. Ces tâches font très souvent appel à des ressources communes tel que l'homme et/ou la machine.

Prenons l'exemple de la Figure 1 modélisant un procédé comme un SED où une souris se déplace de manière spontanée à l'intérieur d'un labyrinthe. Les salles S_i, où i ∈ {0, 1, 2}, communiquent par des portes unidirectionnelles E_1 et E_3 et bidirectionnelle E_2 et e_i définit l'événement : « la souris passe par la porte E_i ».



a) Procédé b) Modèle discret du Procédé

Figure 1 : Exemple de la souris dans un labyrinthe

Soit Σ l'ensemble des événements : Σ = {e_1, e_2, e_3}. Les situations possibles correspondent à la présence de la souris dans la salle S_0, S_1 ou S_2. Ceci définit trois états différents, relatifs aux possibilités d'occupation des salles. Le passage de l'état S_0 « souris en S_0 » à l'état S_1 « souris en S_1 » a lieu suite à l'occurrence de l'événement e_1 « la souris passe par la porte E_1 ». Le modèle discret du procédé peut être construit (Figure 1b) et l'espace d'états s'écrit alors X = {S_0, S_1, S_2}.

La plupart des SED présente des caractéristiques communes telles que :

- Le parallélisme : De nombreux événements peuvent se dérouler simultanément et indépendamment dans diverses parties du système.
- La synchronisation : L'accomplissement de certains événements nécessite la disponibilité simultanée de plusieurs ressources ou la vérification simultanée de plusieurs conditions. La fin d'un événement peut entraîner l'apparition simultanée de plusieurs autres événements.
- La concurrence : Certains événements excluent l'apparition simultanée d'autres événements. Par exemple, certains procédés ne permettent pas de traiter plus d'une pièce à la fois.

De nombreux outils permettent la modélisation des SED. La représentation des SED dépend essentiellement de la granularité de modélisation et de l'objectif recherché. Parmi ces outils, on peut citer les automates à états finis, le GRAFCET, les réseaux de Petri, …. Nous détaillons dans un premier temps, le modèle de base de la modélisation des SED à savoir les automates à états finis. Nous présentons dans un second temps les autres outils de modélisation habituellement retrouvés pour la modélisation des SED.

1. Les automates à états finis

Les automates permettent de traiter mathématiquement les problèmes relatifs aux SED, essentiellement d'un point de vue logique. La théorie des automates à états finis a été principalement développée avec la théorie des langages. Ces modèles permettent de spécifier

les ensembles d'états et de transitions entre ces états [ARN 92]. Leur utilisation réside autant dans la description de la PO que dans la spécification de la commande. En effet, ils décrivent le comportement normal et/ou anormal d'un procédé de manière asynchrone et permettent une observation détaillée du procédé. Cependant, la granularité de la modélisation par automates est un élément déterminant impliquant des problèmes de lisibilité et d'explosion du nombre d'états [GEN 00].

Les automates à états décrivent les SED comme un ensemble d'événements Σ associé à un ensemble d'états X. Ces événements font évoluer le procédé d'un état à un autre. Les SED sont basés sur l'alphabet d'un langage où les séquences d'événements sont des mots de ce langage [RAM 89b]. On appelle « automate » un dispositif qui engendre un langage en manipulant l'alphabet (les événements). Un automate est défini par un 5-uplet $G = (X, \Sigma, \delta, x_0, X_m)$ tel que :

- X est l'ensemble fini des états qui constitue l'espace discret des états,
- Σ est un alphabet fini décrivant l'ensemble des événements,
- δ est la fonction de transition d'état $\delta : X \times \Sigma \rightarrow X$,
- x_0 est l'état initial représenté par une flèche,
- X_m est l'ensemble des états marqués qui définissent les états finaux, $X_m \subseteq X$.

Les automates à états sont basés sur des notions de langages et d'événements permettant la manipulation d'algorithmes mathématiques. Nous présentons quelques rappels sur la formalisation des langages et des automates à états.

1.1. Notion de Langages

Un langage L est un ensemble de mots, ou de chaînes, formés à partir des événements défini sur un alphabet Σ. La longueur d'un mot *s*, notée |*s*|, correspond au nombre d'événements qui le composent. Soit Σ = {α, β, γ} un alphabet, un langage de cet ensemble peut être L_1 = {ε, α, αββ} constitué de 3 mots où le mot vide est noté ε, soit |ε| = ∅.

Un langage L engendré par un automate d'états fini *G* est dit régulier s'il inclut toutes les traces qui peuvent être exécutées à partir de l'état initial de *G*. Il est noté L(*G*) et est définit par :

$$L(G) = \{s \in \Sigma^* | \ \delta(x_0, s) \neq \varnothing\}$$

où Σ^* est l'ensemble des séquences d'événements de longueur finie.

Une chaîne s_1 est dite préfixe d'une chaîne s_3 s'il existe une chaîne $s_2 \in \Sigma^*$ telle que $s_1 s_2 = s_3$. Dès lors ; une chaîne est préfixe d'elle-même. Un langage $L \subseteq \Sigma^*$ est dit préfixe clos, et noté *pr*(L), s'il comprend tous les préfixes des chaînes de L.

$$pr(L) = \{s_1 \in \Sigma^* | \exists s_2 \in \Sigma^*, s_1 s_2 \in L\}.$$

Un langage *L* est *fermé par ses préfixes* (ou *préfixe clos*) si *L* = *pr*(L). Par exemple, si Σ = {α, β} et pour L = {ε, α, αβ, αββ}, la chaîne αβ est un préfixe de la chaîne αββ alors L = *pr*(L). Par contre, L = {ε, α, αββ}n'est pas préfixe clos, d'où L ≠ *pr*(L).

On définit l'ensemble des traces bloquantes de L comme les traces n'ayant plus d'extensions dans L, c'est-à-dire, $s \in L$ est une trace bloquante si $\{s\}\Sigma^* \cap L = \{s\}$. Un état atteint par une trace bloquante est alors appelé état bloquant.

Le langage marqué $L_m(G)$ constitue tous les mots de l'automate *G* dont l'exécution traduit la réalisation d'une tâche. En effet, pour un automate fini $G = (X, \Sigma, \delta, x_0, X_m)$, l'ensemble des chaînes $s \in \Sigma^*$, telles que $\delta(x_0, s) \in X$, définit le langage généré par *G* et est noté L(G). Si maintenant $s \in L(G)$ telles que $\delta(x_0, s) \in X_m$, alors le langage est dit marqué par *G* et est noté $L_m(G)$. Par conséquent, $L_m(G) \subseteq L(G)$ pour $L(G) \subseteq \Sigma^*$ et pour L(G) non vide. Dès lors, L(G) est dit préfixe clos et contient $L_m(G)$ qui est défini par :

$$L_m(G) = \{s \in L(G) : \delta(x_0, s) \in X_m\}$$

On appelle langage de spécification K, le comportement désiré d'un système du procédé, où K est un sous langage de L. Ce comportement K est généralement modélisé sous forme d'automates décrivant les contraintes à imposer en termes de séquences d'événements possibles et des états à interdire. Cependant, l'existence d'événements non commandables implique que l'on ne peut pas restreindre le comportement L d'un SED à n'importe quel sous langage K de L.

Un langage K est dit commandable par rapport au langage L(G) lorsque toute occurrence d'un événement non commandable physiquement possible entraîne une évolution vers un état appartenant également au langage désiré K.

1.2. Caractéristiques d'un automate

Plusieurs caractéristiques d'un automate peuvent être exprimées :

- Un automate est dit déterministe si l'état initial est unique et si la relation de transition, appliquée à un couple (x_i, σ), définit toujours un unique état.
- Un état $x_i \in X$ est dit accessible s'il existe une chaîne $s \in \Sigma^*$ pouvant y accéder depuis l'état initial telle que $\delta(x_0, s) = x_i$.
- Un état $x_i \in X$ est dit co-accessible s'il existe une chaîne $s \in \Sigma^*$ permettant d'atteindre un état marqué telle que $\delta(x_i, s) = x_j \in X_m$.
- Un automate *G* est qualifié de non bloquant lorsque tout état accessible est co-accessible. En d'autres termes un automate est non bloquant s'il existe une chaîne *s* générée par *G* permettant de retourner à l'état initial ou d'atteindre un état marqué.

La modélisation des systèmes complexes est très difficile à cause du nombre d'états qui augmente au fur et à mesure. On parle alors d'explosion combinatoire des états et transitions. C'est un problème récurrent à la modélisation des SED par automate à états.

1.3. Notion d'événements

L'ensemble des événements Σ peut être décomposé selon la notion d'observabilité. Ainsi, les événements liés aux capteurs et actionneurs d'un procédé pouvant être observés appartiendront à l'ensemble des événements observables $\Sigma_o \subseteq \Sigma$ alors que les événements de défauts ou internes seront considérés comme non observables $\Sigma_{uo} \subseteq \Sigma$.

Une caractéristique importante pour un procédé est de pouvoir distinguer les événements qui sont commandables, ou contrôlables, de ceux qui ne le sont pas. Les événements, dans les procédés réels, ne sont pas toujours générés de façon spontanée, mais réagissent à des commandes d'entrée, générant des réponses en sortie. Les interactions entre la PC spécifiée et la PO sont alors mises en évidence d'un point de vue commande. Par conséquent, nous adopterons par la suite que les ordres sont des événements envoyés par la PC et permettent de commander le procédé. Ils sont vu comme des sorties de la PC et sont des événements commandables $\Sigma_c \subseteq \Sigma_o$. A contrario, les événements provenant des capteurs répondent à une action en entrée de la PC et sont donc non commandables $\Sigma_{uc} \subseteq \Sigma_o$ [BAL 93].

1.4. Compositions d'automates

Un procédé est souvent composé de plusieurs éléments en interaction plus ou moins forte les uns avec les autres. Deux machines peuvent travailler ensemble mais disposer de deux automates décrivant leur comportement individuel où chaque automate possède alors son propre alphabet d'événements (Σ_1 et Σ_2). Le comportement de l'ensemble des deux machines peut être décrit par un seul en réalisant une composition des deux automates. La composition, ou produit croisé, est l'outil permettant de construire un procédé complexe à partir de procédés élémentaires. Différentes compositions existent :

- La composition synchrone d'événements en commun ($\Sigma_1 \cap \Sigma_2 \neq 0$)
- La composition asynchrone où aucun événement n'est en commun ($\Sigma_1 \cap \Sigma_2 = 0$)
- La composition synchrone totale où tous les événements sont en commun ($\Sigma_1 = \Sigma_2$)

1.4.1. Composition synchrone

Soit $L_1 \subseteq \Sigma^*_1$ et $L_2 \subseteq \Sigma^*_2$, avec $\Sigma_1 \cap \Sigma_2 \neq 0$ et $\Sigma = \Sigma_1 \cup \Sigma_2$

La composition synchrone de deux automates $G_1 = (X_1, \Sigma_1, \delta_1, x_{01}, X_{1m})$ et $G_2 = (X_2, \Sigma_2, \delta_2, x_{02}, X_{2m})$ noté $G_1||G_2$ est définie par le 5-uplet :

$(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \delta_1 \times \delta_2, x_{01} \times x_{02}, X_{1m} \times X_{2m})$

où $X_1 \times X_2$ est l'ensemble des états définit par le produit cartésien de X_1 par X_2,

$\Sigma = \Sigma_1 \cup \Sigma_2$ représente l'alphabet de $G_1||G_2$,

$x_{01} \times x_{01}$ est l'état initial,

$X_{1m} \times X_{2m}$ est l'ensemble des états marqués

et où la fonction de transition d'états $\delta_1 \times \delta_2$ est définie par :

$(\delta_1 \times \delta_2)((x_1, x_2), \boldsymbol{\sigma})=(x_1', x_2')$ si $\delta_1(x_1, \boldsymbol{\sigma})= x_1'$ et $\delta_2(x_2, \boldsymbol{\sigma})= x_2'$ (synchronisation) (1)

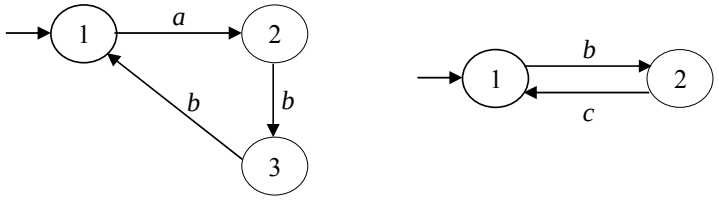
$(\delta_1 \times \delta_2)((x_1, x_2), \boldsymbol{\sigma})=(x_1', x_2)$ si $\delta_1(x_1, \boldsymbol{\sigma})= x_1'$ avec $\boldsymbol{\sigma} \in \Sigma_1 \setminus \Sigma_2$ (2)

$(\delta_1 \times \delta_2)((x_1, x_2), \boldsymbol{\sigma})=(x_1, x_2')$ si $\delta_2(x_2, \boldsymbol{\sigma})= x_2'$ avec $\boldsymbol{\sigma} \in \Sigma_2 \setminus \Sigma_1$ (3)

où $\boldsymbol{\sigma} \in \Sigma_1 \setminus \Sigma_2$ représente un événement appartenant à l'ensemble Σ_1 mais pas à celui de Σ_2. Les événements appartenant à $((\Sigma_1 \setminus \Sigma_2) \cup (\Sigma_2 \setminus \Sigma_1))$ n'introduisent pas de contrainte et peuvent être exécutés dès que possible.

Prenons l'exemple où $\Sigma_1 = \{a, b\}$ et $\Sigma_2 = \{b, c\}$. Les automates G_1 et G_2 associés aux deux

procédés considérés sont représentés en Figure 2. On en déduit $L_1 = \{\varepsilon, a, ab, abb, \ldots\}$ et $L_2 = \{\varepsilon, b, bc, \ldots\}$ avec $\Sigma_1 \cap \Sigma_2 = b$ et $\Sigma = \Sigma_1 \cup \Sigma_2 = \{a, b, c\}$.



a) Automate G_1 b) Automate G_2

Figure 2 : Automates G_1 et G_2 avant composition synchrone

Dans leur état initial, les deux automates ne sont pas sensibles à l'événement commun *b* ; la relation (1) ne s'applique pas encore. A partir de l'état initial (1,1) de la composition, l'événement *a* peut être exécuté car seule la relation (2) s'applique ici, l'automate global se trouve alors en état (2,1). L'automate G_1 évolue depuis son état initial vers l'état suivant alors que l'automate G_2 est toujours dans son état initial. Ces deux automates deviennent alors sensibles à l'événement commun *b*. La relation (1) peut alors s'appliquer et fait évoluer l'automate de composition dans l'état (3,2). L'automate de composition synchrone $G_1||G_2$ est obtenu par construction et modélise le comportement global représenté en Figure 3.

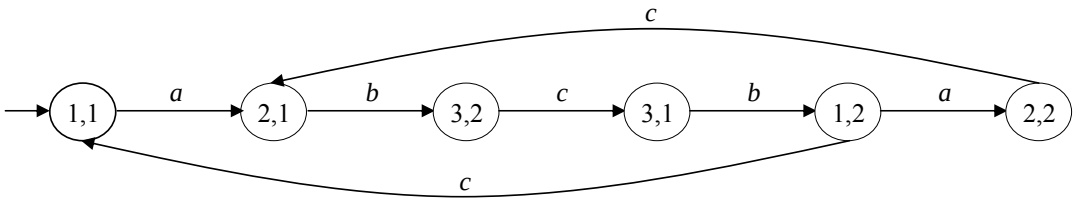


Figure 3 : Automate de composition synchrone de G_1 || G_2

1.4.2. Composition asynchrone

Soit $L_1 \subseteq \Sigma^*_1$ et $L_2 \subseteq \Sigma^*_2$, avec $\Sigma_1 \cap \Sigma_2 = 0$ et $\Sigma = \Sigma_1 \cup \Sigma_2$ (pas d'événement commun).

La composition asynchrone de deux automates $G_1 = (X_1, \Sigma_1, \delta_1, x_{01}, X_{1m})$ et $G_2 = (X_2, \Sigma_2, \delta_2, x_{02}, X_{2m})$ noté $G_1||G_2$ est définie par le 5-uplet :

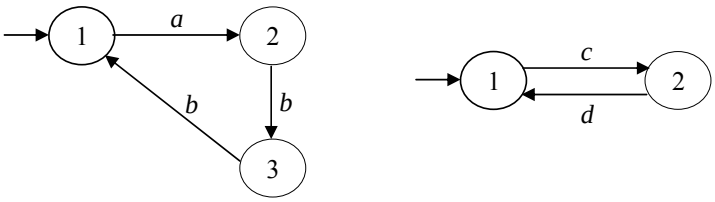
$(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \delta_1 \times \delta_2, x_{01} \times x_{02}, X_{1m} \times X_{2m})$ avec

$$(\delta_1 \times \delta_2)((x_1, x_2), \boldsymbol{\sigma}) = (x_1', x_2) \text{ si } \delta_1(x_1, \boldsymbol{\sigma}) = x_1' \text{ pour } \boldsymbol{\sigma} \in \Sigma_1 \quad (4)$$

$$(\delta_1 \times \delta_2)((x_1, x_2), \boldsymbol{\sigma}) = (x_1, x_2') \text{ si } \delta_2(x_2, \boldsymbol{\sigma}) = x_2' \text{ pour } \boldsymbol{\sigma} \in \Sigma_2 \quad (5)$$

La composition asynchrone ne représente qu'un cas particulier de la composition synchrone.

Prenons l'exemple de la Figure 4 où deux automates G_1 et G_2 sont représentés avec $\Sigma_1 = \{a, b\}$ et $\Sigma_2 = \{c, d\}$. On en déduit $L_1 = \{\varepsilon, a, ab, abb, \ldots\}$ et $L_2 = \{\varepsilon, c, cd, \ldots\}$ avec $\Sigma_1 \cap \Sigma_2 = 0$ et $\Sigma = \Sigma_1 \cup \Sigma_2 = \{a, b, c, d\}$.



a) Automate G_1 b) Automate G_2

Figure 4 : Automates G_1 et G_2 avant composition asynchrone

A partir de leur état initial, les deux automates peuvent évoluer soit par l'événement *a* avec la relation (4), soit par l'événement *c* par la relation (5). A partir de son état initial (1,1), la composition asynchrone fait évoluer l'automate soit en état (2,1), soit en état (1,2). L'automate de composition asynchrone $G_1||G_2$ modélisant le comportement global est représenté en Figure 5.

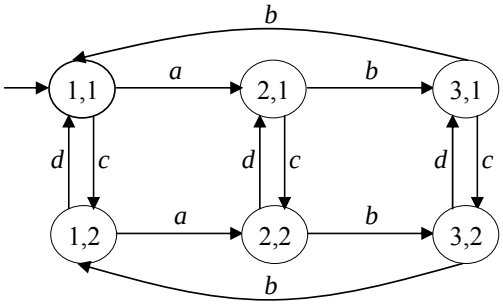


Figure 5 : Automate de composition Asynchrone de G_1 || G_2

On peut remarquer que si l'on définit *m* le nombre d'états associé à l'automate G_1 et *n* le nombre d'états associés à l'automate G_2, alors la composition asynchrone de $G_1||G_2$ possède au final $m \times n$ états.

1.4.3. Composition synchrone totale

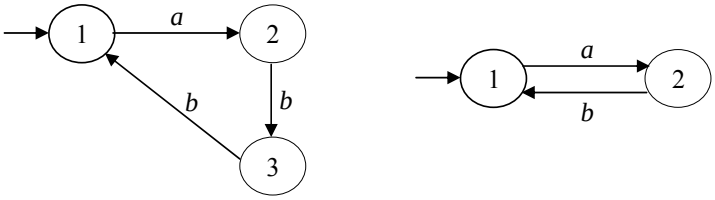
La composition synchrone totale, ou complète, part de l'hypothèse où $\Sigma = \Sigma_1 = \Sigma_2$. A partir de là, les relations (2) et (3) ne peuvent plus être vérifiées et la relation de composition se réduit à :

$$(\delta_1 \times \delta_2)((x_1,x_2), \boldsymbol{\sigma}) = (x_1',x_2') \text{ si } \delta_1(x_1,\boldsymbol{\sigma}) = x_1' \text{ et } \delta_2(x_2,\boldsymbol{\sigma}) = x_2' \qquad (6)$$

pour $\boldsymbol{\sigma} \in \Sigma$ et $(x_1,x_2) \in (X_1 \times X_2)$

Cette opération, généralement notée "$G_1 \times G_2$", débouche sur des situations de blocage lorsque l'on ne peut plus évoluer sur des événements communs à partir de chaque état des procédés [FER 04].

Prenons l'exemple de deux automates G_1 et G_2 avec $\Sigma_1 = \Sigma_2 = \{a, b\}$. On en déduit $L_1 = \{\varepsilon, a, ab, abb, \ldots\}$ et $L_2 = \{\varepsilon, a, ab, \ldots\}$ et $\Sigma = \Sigma_1 = \Sigma_2 = \{a, b\}$ (Figure 6).



a) Automate G_1 b) Automate G_2

Figure 6 : Automates G_1 et G_2 avant composition synchrone totale

De leur état initial (1,1), les deux automates peuvent évoluer par l'événement commun *a* avec la relation (6) pour se retrouver en état (2,2), puis par l'événement commun *b* pour atteindre l'état (3,1). A partir de cet état, il n'existe plus d'événement qui peut faire évoluer l'automate, il s'agit par conséquent d'un état bloquant. L'automate de composition synchrone total $G_1||G_2$ est représenté Figure 7.

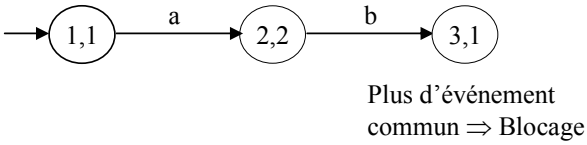


Figure 7 : Automate de composition synchrone totale de G_1 || G_2

Propriété 1 : La composition synchrone "||" est commutative et associative [FER 04]. Soit $L_1 \subseteq \Sigma^*_1$, $L_2 \subseteq \Sigma^*_2$ et $L_3 \subseteq \Sigma^*_3$ alors :

$$L_1||(L_2||L_3) = (L_1||L_2)||L_3 = L_1||L_2||L_3.$$

$$L_1||L_2||L_3 = L_2||L_3||L_1 = L_3||L_1||L_2$$

1.4.4. Projection naturelle

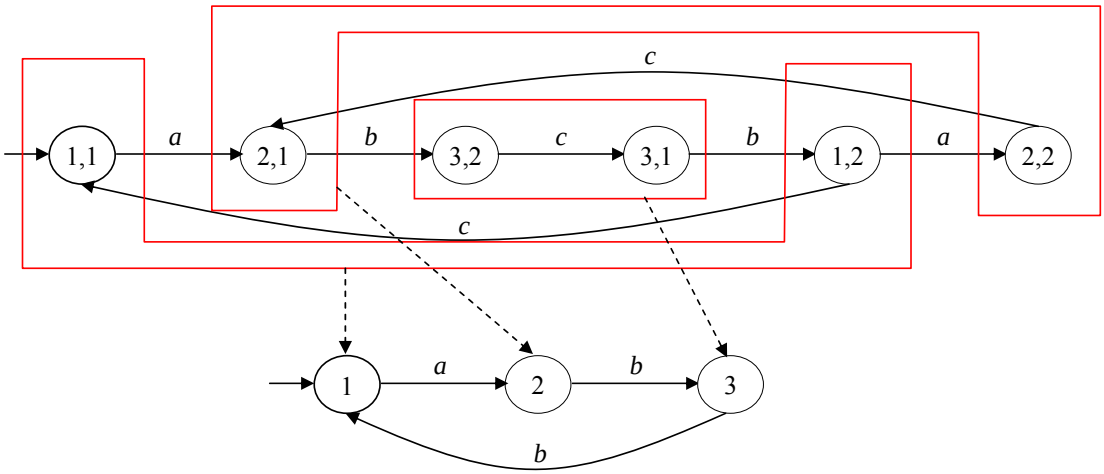
La composition synchrone est généralement définie sur les langages L_i et nécessite des opérations de projection et de projection inverse. La projection naturelle est un outil permettant d'obtenir, à partir d'automate, une décomposition d'automates en ne considérant que les événements observables. Elle "efface" tous les événements non observables présent dans une chaîne d'événements [SAM 95].

Prenons un automate G_1 de langage $L_1 \subseteq \Sigma_1^*$ avec $\Sigma_1 = \Sigma_{o1}$ l'ensemble des événements l'ensemble des événements observables sur l'automate G_1 et un automate G_2 de langage $L_2 \subseteq \Sigma_2^*$ avec $\Sigma_2 = \Sigma_{o2}$ l'ensemble des événements observables sur l'automate G_2. L'automate global G est obtenu par composition synchrone entre l'automate G_1 et G_2 : $G = G_1 \,||\, G_2$. Par conséquent, G est un automate de langage L, où $L_1 \subseteq L$ et $L_2 \subseteq L$, avec $\Sigma = \Sigma_1 \cup \Sigma_2$ l'ensemble des événements et $\Sigma_o = \Sigma_{o1} \cup \Sigma_{o2}$ est l'ensemble des événements observables sur l'automate G.

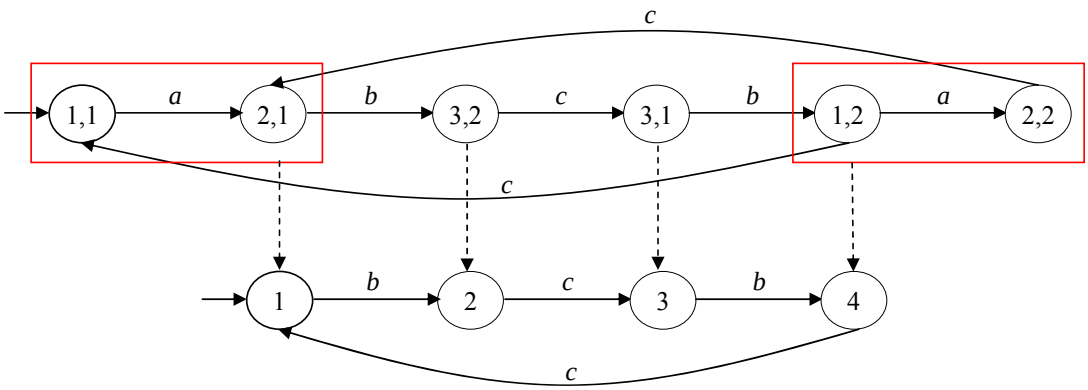
La projection naturelle sur le langage L_1, notée "$P_{L1} : \Sigma_1^* \rightarrow \Sigma_{o1}^*$", de l'automate G permet d'obtenir l'automate G_1. Elle consiste à ne prendre sur G que les événements de L_1 comme observables. Dès lors, tous les autres événements ne sont pas considérés. La Figure 8a montre un exemple d'obtention d'un automate G_1 de projection naturelle P_{L1} à partir d'un automate G à 6 états. Pour l'ensemble des événements $\Sigma = \{a, b, c\}$ de G, la projection P_{L1} consiste à

prendre uniquement les événements *a* et *b* comme observables. Il en ressort un automate G_1 à 3 états.

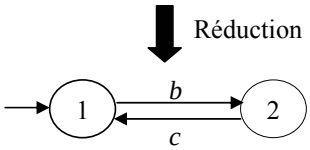
De la même manière, la projection naturelle sur L_2, notée "$P_{L2} : \Sigma_2^* \rightarrow \Sigma_{o2}^*$", de l'automate *G* est l'automate G_2 de langage L_2. Cette fois-ci, ce sont les événements de L_2 sur *G* qui sont considérés comme observables. La Figure 8b montre l'obtention d'un automate G_2 de projection naturelle P_{L2} à partir d'un automate *G* à 6 états. Pour l'ensemble des événements $\Sigma = \{a, b, c\}$ de *G*, la projection P_{L2} consiste à prendre uniquement les événements *b* et *c* comme observables. Il en ressort un automate G_2 à 4 états qui peut être réduit à 2 états.



a) Automate G_1 par la projection P_{L1} de G



b) Automate G_2 par la projection P_{L2} de G



c) Automate équivalent de G_2

Figure 8 : Exemple de projection naturelle

La projection $P_L : \Sigma^* \rightarrow \Sigma_o^*$ est définie par :

- $P_L(\varepsilon) = \varepsilon$
- $P_L(\sigma) = \sigma$ si $\sigma \in \Sigma_o^*$
- $P_L(\sigma) = \varepsilon$ si $\sigma \in \Sigma_{uo}^*$ et $\sigma \notin \Sigma_o^*$

- $P_L(s\sigma) = P_L(s)P_L(\sigma)$ avec s et $\sigma \in \Sigma^*$

La projection d'un automate G est composé de l'ensemble de ses événements observables d'où : $P_L(G_1) : (\Sigma_1 \cup \Sigma_2)^* \rightarrow \Sigma_1^*$ et $P_L(G_2) : (\Sigma_1 \cup \Sigma_2)^* \rightarrow \Sigma_2^*$

Basé sur la projection naturelle, l'opération de composition synchrone peut être retrouvée par l'opérateur de projection inverse P_L^{-1} défini par :

$$P_L^{-1}(u) = \{s \in L : P_L(s) = u\}$$

où la projection inverse d'une chaîne *u* consiste à prendre en compte les événements non observables appartenant au langage L et présent dans la chaîne *s*.

Soit Σ_1, Σ_2 deux ensembles d'événements où $\Sigma = \Sigma_1 \cup \Sigma_2$ où $P_{L1} : \Sigma_1^* \rightarrow \Sigma_{o1}^*$ est la projection naturelle de Σ_1 et $P_{L2} : \Sigma_2^* \rightarrow \Sigma_{o2}^*$ la projection naturelle de Σ_2. Alors pour une paire de langage $L_1 \subseteq \Sigma^*_1$ et $L_2 \subseteq \Sigma^*_2$, le produit synchrone de L_1 et L_2 est définit comme l'intersection de la projection inverse P_{L1}^{-1} de L_1 et de la projection inverse P_{L2}^{-1} de L_2. En d'autres termes, $L_1||L_2 = \{s \in \Sigma^* | P_{L1}(s) \in L_1 \ \& \ P_{L2}(s) \in L_2\}$.

$$L_1||L_2 = P_{L1}^{-1}(L_1) \cap P_{L2}^{-1}(L_2)$$

Si nous interprétons la composition synchrone "||" comme étant la construction d'un modèle global à partir d'automates locaux, alors la projection naturelle "P_L" correspond à la construction de modèles locaux à partir d'un modèle global.

2. Les autres outils de modélisation

Les SED peuvent également être modélisés par d'autres outils de représentations graphiques tel que le GRAFCET, les réseaux de Petri et les *Statecharts* [CAL 90]. Chacun d'entre eux a été développé plus ou moins pour un objectif de modélisation de la commande afin de décrire le comportement désiré d'un procédé. Cependant, certains d'entre eux, comme les réseaux de Petri ou les *Statecharts*, ont vu leur utilisation bifurquer vers le diagnostic des SED en considérant le fait que tout ce qui n'est pas normal est anormal ; et donc défaillant. Nous allons présenter rapidement ces quelques outils afin de comprendre leur place dans les approches de diagnostic.

2.1. GRAFCET

La conception d'un SAP nécessite des outils permettant de matérialiser la pensée technique en conception, d'aider à la réalisation mais aussi d'assister son exploitation. Le GRAFCET (GRAphe Fonctionnel de Commande Etape/Transition) est aujourd'hui couramment employé pour la spécification de la commande des SED en raison de son ergonomie et de sa capacité à représenter le comportement dynamique. C'est un outil de modélisation, de spécification et de conception qui permet de réaliser une représentation graphique du système à commander et de décrire les fonctionnalités d'un automatisme [DAV 89], [LHO 97] et [CAR 02]. Le GRAFCET n'est cependant adapté qu'aux systèmes de nature séquentielle dont les entrées/sorties sont réduites à des variables booléennes [GEN 00]. Les possibilités offertes pour exprimer le parallélisme et la synchronisation ainsi que le concept de réceptivité en font un outil adapté pour décrire simplement la commande des systèmes logiques parallèles et/ou synchronisés.

C’est la norme internationale IEC 60848 [IEC 02] qui définit le GRAFCET pour la description du comportement de la partie séquentielle des systèmes de commande. La Figure 9 présente la constitution d’un GRAFCET qui s’articule autour :

- d’éléments graphiques de base : les étapes, les transitions et les liaisons orientées,
- d’une interprétation associant les expressions logiques : les actions associées aux étapes et les réceptivités associées aux transitions,
- de règles d’évolution définissant le comportement dynamique d’une commande.

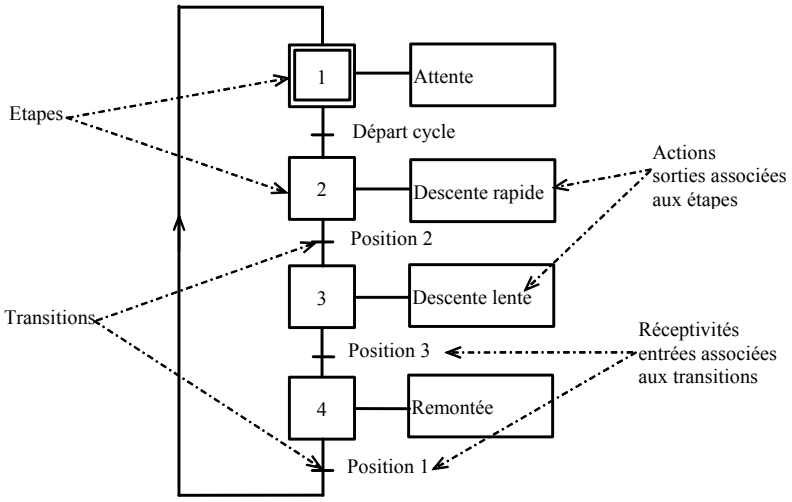


Figure 9 : Composition d’un GRAFCET

L’autre avantage du GRAFCET se traduit par sa faculté à être implémenté facilement dans un Automate Programmable Industriel (API). En effet, les API acceptent différents langages de programmation dont le "Sequential Function Chart" (SFC) qui correspond à un langage d’implémentation équivalent au modèle GRAFCET lui donnant ainsi un caractère opérationnel [IEC 93].

2.2. Réseaux de Petri

Les Réseaux de Petri (RdPs) [PET 81] constituent un outil de modélisation particulièrement adapté pour spécifier le comportement des systèmes industriels, des réseaux de communication et des bases de données reparties. Les RdPs sont représentés autour d’un langage graphique et d’un langage mathématique [MUR 89]. Un RdP se présente sous forme de places et de transitions, reliées par des arcs. Une place peut contenir un nombre entier de marques communément appelés jetons (Figure 10).

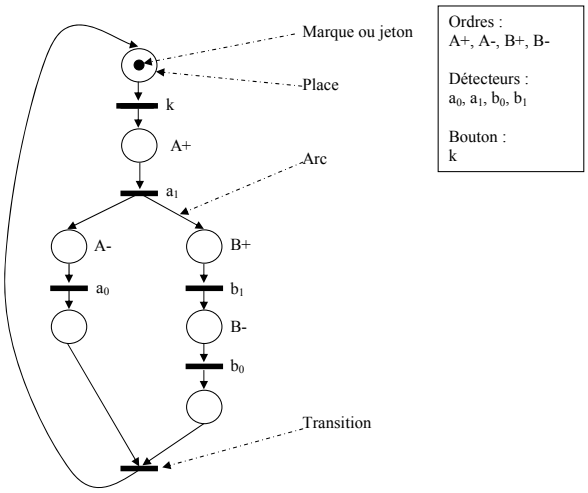


Figure 10 : Constitution d'un Réseau de Petri

L'état d'un RdP est représenté par son marquage qui est le vecteur courant du nombre de jeton dans l'ensemble de ses places. A tout marquage accessible, obtenu à partir du marquage initial par le franchissement d'une séquence de transitions, correspond un état du système. Formellement, un RdP est défini par un quintuplet $Q = (P, T, Pré, Post, m_0)$ avec :

- $P = \{P_1, P_2, \ldots, P_n\}$: l'ensemble des places,
- $T = \{T_1, T_2, \ldots, T_m\}$: l'ensemble des transitions,
- $Pré : P \times T \rightarrow \aleph$: la matrice d'incidence avant,
- où $\aleph$ est l'ensemble des entiers non négatifs
- $Post : P \times T \rightarrow \aleph$: la matrice d'incidence arrière,
- $m_0 : P \rightarrow \aleph$: le marquage initial du réseau.

Cet outil permet de modéliser et de visualiser des primitives de comportement telles que la synchronisation, le parallélisme, le partage de ressources ou le séquencement [BRA 83] et [DAV 89]. Ils permettent également la représentation des informations quantitatives car les marques peuvent être accumulées dans les places. Les RdPs sont tout aussi bien adaptés à la spécification, à la conception qu'à la simulation des procédés automatisés pour la PC et/ou la PO. C'est un outil non normalisé qui possède de nombreuses déclinaisons comme les RdPs colorés, temporisés ou stochastiques… Cette richesse d'extensions fait que les RdPs peuvent s'appliquer à la plupart des phases de développement d'un système, de la spécification de la commande à la supervision en passant par le diagnostic et la validation.

Un certain nombre de propriétés des systèmes peuvent être étudiées comme la vivacité, le conflit ou la bornitude. Un RdP est dit vivant lorsque aucune transition ne peut devenir infranchissable. Un blocage décrit une transition ne pouvant plus être franchie. Un conflit apparaît dans le cas de partage d'une ressource commune quand une même place est en amont de deux ou plusieurs transitions. Enfin, un RdP est dit borné lorsqu'il possède un nombre fini de marques. Les méthodes de recherche de propriétés sont basées sur l'élaboration du graphe de marquage et l'algèbre linéaire [DAV 89].

Les RdPs permettent l'analyse des systèmes avec des comportements asynchrones, parallèles, non déterministes et distribuées. Ils peuvent également décrire des comportements

de défaut et s'exposent alors à leur principal désavantage qu'est l'explosion combinatoire pour la modélisation des procédés complexes.

2.3. Statecharts ou Diagramme d'états

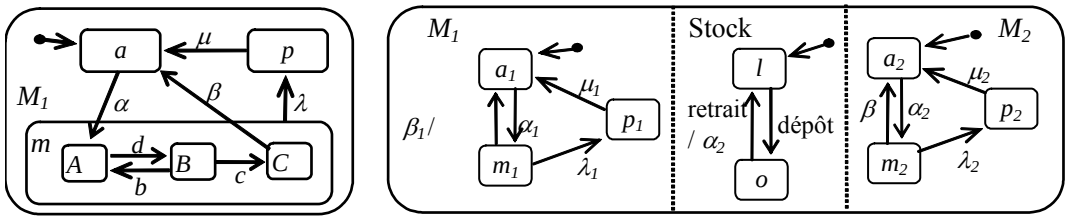
Le *Statecharts* (ou diagrammes d'états) est un formalisme de description graphique défini par Harel [HAR 87] adaptés à la spécification des systèmes réactifs complexes. Il puise ses concepts dans ceux des machines d'états et ceux des higraphs (hiérarchie). Il décrit tous les états dans lesquels est susceptible de se trouver une entité ou une interaction, ainsi que les transitions possibles entre ces états [DAS 03a]. Il est habituellement caractérisé par une expression liant les propriétés d'un *Statecharts* :

Statecharts = automates + profondeur + orthogonalité + communication par diffusion

Les automates comportent des états et des transitions. Les transitions entre états sont étiquetées par des expressions de la forme "*événement déclenchant (condition d'autorisation) / action*" ; l'action, qui a une durée nulle, est synchrone avec l'événement déclenchant. Dans la Figure 11b, on peut voir que pour la machine M_1, le passage du sous état a_1 vers le sous état m_1 est effectué par l'événement déclenchant α_1. L'événement β_1 permet le retour en état a_1 et conditionne l'autorisation de l'action *Dépôt*. Des actions peuvent également être associées à l'activation des états et à leur désactivation ainsi que des activités qui s'exécutent tant que l'état concerné reste actif.

Une particularité du formalisme des *Statecharts* est qu'il permet de décomposer un même état selon une ou plusieurs sous machines d'états concurrentes. Il définit le comportement hiérarchisé des fonctions. La notion de profondeur (hiérarchie) sert à décomposer des super états en sous états, dessinés à l'intérieur du super état. Cette notion permet l'agrégation d'états pour simplifier la représentation des graphes dans une approche ascendante et la décomposition par niveaux d'abstraction dans une approche descendante. Ainsi, la Figure 11a représente un modèle plus affiné d'une machine avec son état *m* décomposé en 3 sous états pour indiquer que l'état de marche commence par l'activation du sous état *A*, suivi d'une alternance entre les deux sous états *A* et *B* jusqu'à l'occurrence de l'événement *c* qui active le sous état *C*. La transition étiquetée par l'événement λ quitte le super état *m*. Cet événement λ représente un défaut et indique que l'arrivée d'une panne à partir de n'importe quel sous état de *m* entraîne la désactivation de ce sous état et de l'état *m* pour activer l'état *p*. Ainsi cette transition remplace 3 transitions partant, chacune, de l'un des sous états de *m*.

L'orthogonalité, correspondant à la possibilité de décomposer un état en sous états actifs simultanément, se représente en séparant les sous états à l'aide d'une ligne pointillée. Cette notion permet de diminuer le nombre d'états manipulés en considérant des automates localement indépendants. Afin d'éviter les problèmes temporels d'évolution, les événements sont émis en diffusion et donc disponibles partout au même instant. Les notions d'orthogonalité et de communication sont illustrées par la Figure 11b qui présente deux machines synchronisées par un stock à deux états : libre (*l*) et occupé (*o*). L'événement β_1, indiquant la fin de l'opération sur M_1, entraîne la diffusion de l'action dépôt qui est utilisée simultanément au niveau de l'automate de stock en tant qu'événement pour activer l'état *o*. De même, le retrait du stock (événement retrait sur la transition sortante du sous état *o*) entraîne la diffusion de l'action α_2 qui est utilisée en tant qu'événement associé à la transition d'activation de l'état m_2 de la machine M_2.



a) Profondeur b) Orthogonalité et communication

Figure 11 : Illustration des propriétés des Statecharts

3. Quel outil pour quel modèle ?

Les différents outils que nous venons de citer disposent d'avantages et d'inconvénients en fonction de l'objectif désiré et du fonctionnement, normal ou anormal, que l'on souhaite modéliser. En effet, certains outils sont mieux adaptés que d'autres pour réaliser la fonction demandée. De plus, la modélisation d'un système pour réaliser une commande ne possède pas la même caractéristique et finesse qu'un modèle pour le diagnostic.

Les RdPs permettent de modéliser les systèmes à événements discrets de n'importe quelle nature. Leur utilisation consiste essentiellement à décrire le comportement souhaité, ce qui en fait un outil performant dans le cadre de la synthèse de la commande. Dans la littérature, des méthodes de diagnostic à base de RdPs existent, les RdPs décrivant alors le comportement normal et défaillant d'un système. L'utilisation de RdPs temporel permet également d'intégrer des contraintes temporelles sur le procédé. Ils sont notamment efficaces pour la représentation de procédés distribués gérant des ressources communes et les événements concurrentiels. Les RdPs ont une tendance à l'explosion combinatoire moins importante que celle des automates à états [CAS 99]. Cependant, ils ne possèdent pas de langage formel, ce qui implique une impossibilité à utiliser les outils de composition et de projection.

Le GRAFCET est destiné à la spécification des automatismes logiques et à la description de leur cahier des charges. Bien que ce soit un modèle inspiré des RdPs, la connaissance de ceux-ci n'est pas nécessaire à sa compréhension et à son utilisation. Cependant, le GRAFCET n'est pas utilisé pour le diagnostic ou la description de PO car il serait un modèle semblable aux automates à états. L'apport essentiel du GRAFCET par rapport aux RdPs est une modélisation claire des entrées, des sorties et de leurs relations. GRAFCET et RdPs permettent notamment de caractériser des aspects de parallélisme et de synchronisation d'événements sur des procédés de nature asynchrone.

Les automates à état semblent très bien adaptés pour la description des comportements de la PO car ils conservent une description asynchrone de tout comportement normal et anormal. Ils sont très utilisés pour le diagnostic des défaillances car la représentation des états de défauts est très facile à obtenir. Ils permettent notamment la manipulation de langage formel par les outils de compositions et projections. Cependant, la construction des modèles reste très intuitive et demande une grande connaissance du procédé. De plus, ils tendent très souvent vers un problème d'explosion combinatoire.

Les *Statecharts* sont, quant à eux, une extension des automates à états grâce à la hiérarchisation qu'ils apportent. Ils ne sont généralement là que pour exprimer cette hiérarchisation des procédés et sont très peu utilisés dans le cadre du diagnostic des défaillances [RAY 04] et sont plutôt utilisés comme des automates enrichis.

Dans le cadre du diagnostic des défaillances, nous souhaitons décrire à la fois le comportement désiré de la commande, mais aussi les comportements anormaux du procédé. Au regard des différentes possibilités des outils, notre choix va s'orienter vers la modélisation par automates à états puisqu'ils permettent à la fois la manipulation des outils mathématiques de compositions et de projections, mais également l'utilisation des langages sous-jacents pour formaliser l'analyse des propriétés du diagnostic comme la notion de diagnosticabilité.

Annexe 2

Bibliothèque d'élément de partie opérative

Nous avons vu qu'un système manufacturier est en fait une imbrication d'éléments de partie opérative (EPO). Dans ce paragraphe, nous souhaitons valider une bibliothèque des modèles de ces EPO afin que par la suite, la réalisation du modèle d'un système manufacturier ne soit que le choix des modèles en fonction de leur technologie. Les EPO modélisés dans la bibliothèque sont :

- Vérin simple effet (VSE) piloté par un distributeur pneumatique 3/2 monostable,
- Vérin double effet (VDE) piloté par un distributeur pneumatique 5/2 monostable,
- VDE piloté par un distributeur pneumatique 5/2 bistable,
- VDE piloté par un distributeur pneumatique 5/3 Centre Ouvert (CO),
- VDE piloté par un distributeur pneumatique 5/3 Centre Fermé (CF),
- Moteur électrique à 1 sens de rotation,
- Moteur électrique à 2 sens de rotation avec contacteur.

Pour une meilleure lisibilité, nous préférons ne présenter que les modèles pratiques des actionneurs permettant d'obtenir les modèles des EPO avec 2 détecteurs par composition synchrone de l'automate des détecteurs avec celui de l'actionneur.

1. VSE piloté par un distributeur 3/2 monostable

Le vérin simple effet (VSE) piloté par un distributeur 3/2 monostable (Figure 12a) réalise sa sortie par l'ordre *SO* et possède un retour mécanique pour sa rentrée. Sa modélisation au niveau préactionneur est alors représentée par un automate 2 états pour l'activation et la désactivation de l'ordre (Figure 12b). En effet, c'est un ressort mécanique qui permet le retour du vérin. Le modèle actionneur va comporter au niveau des transitions bouclantes les informations portées par les détecteurs. Lorsque l'ordre *SO* est activé (état 1), il est possible d'avoir la désactivation du détecteur *b* ou l'activation de *a* (Figure 12c).

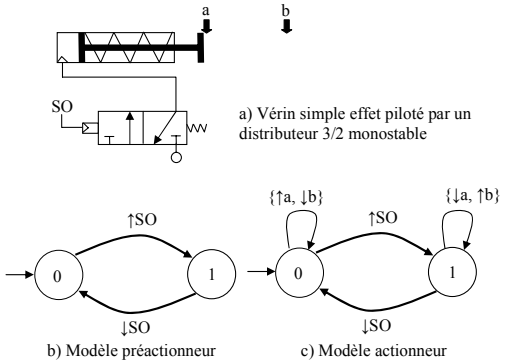


Figure 12 : VSE piloté par distributeur 3/2 monostable

2. VDE piloté par un distributeur 5/2 monostable

Le vérin double effet (VDE) piloté par un distributeur 5/2 monostable (Figure 13) réagit de la même façon que le VSE piloté par un distributeur 3/2 monostable et dispose donc des même modèles préactionneur et actionneur.

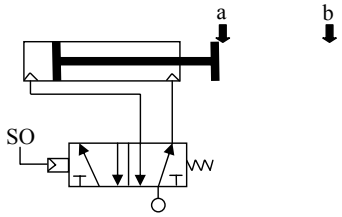


Figure 13 : VDE piloté par distributeur 5/2 monostable

3. VDE piloté par un distributeur 5/2 bistable

Le VDE piloté par un distributeur 5/2 bistable a été modélisé et détaillé précédemment. Il est rappelé ici en Figure 14.

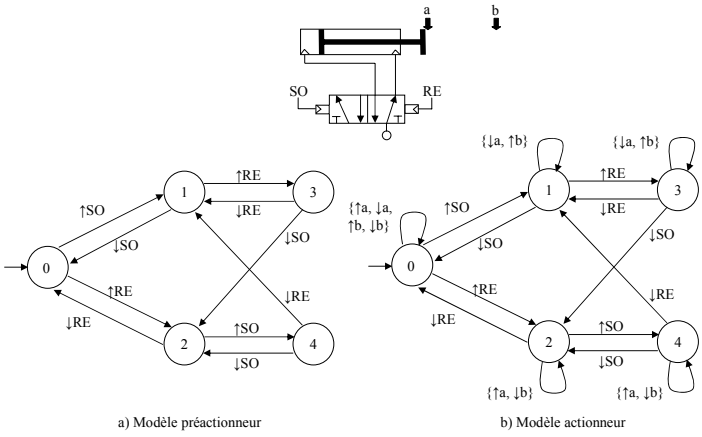


Figure 14 : VDE piloté par distributeur 5/2 bistable

4. VDE piloté par un distributeur 5/3 Centre Ouvert

Le VDE piloté par un distributeur 5/3 Centre Ouvert présenté en Figure 15a se différencie par rapport à l'exemple précédent au niveau de la désactivation des ordres. En effet, la chambre intermédiaire du distributeur permet de garder le vérin dans sa position lors de la désactivation d'un ordre. Bien que la tige du vérin reste dans sa position, il est possible de la déplacer manuellement étant donné que les chambres du vérin ne sont pas compressées.

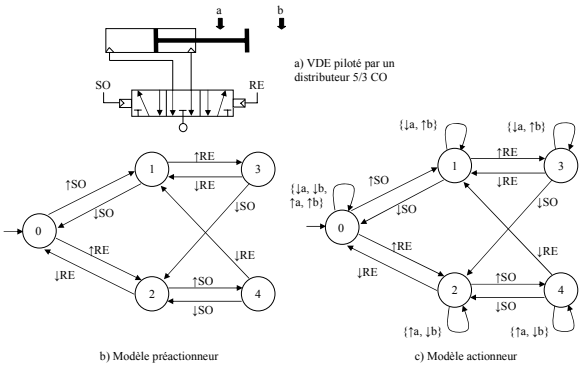


Figure 15 : VDE piloté par distributeur 5/3 CO

Le modèle préactionneur (Figure 15b) est semblable à celui du VDE piloté par un distributeur 5/2 bistable mais le modèle actionneur s'en trouve changé (Figure 15c). En effet, après l'activation de l'ordre *SO* (état 1) et sa désactivation (retour en état 0). Le distributeur retourne en position intermédiaire et le vérin va s'arrêter à sa position actuelle. Si l'on considère qu'un élément extérieur peut déplacer la tige du vérin lorsque aucun ordre est envoyé, alors il est possible d'avoir tous les événements non commandables des détecteurs à l'état 0. Dans le cas où l'on ne considère aucun déplacement possible dans la position intermédiaire du distributeur, alors il n'y aura aucun événement rebouclant à l'état 0.

5. VDE piloté par un distributeur 5/3 Centre Fermé

Le VDE piloté par un distributeur 5/3 Centre Fermé (Figure 16a) présente les mêmes caractéristiques que le distributeur 5/3 CO sauf pour la position de la chambre intermédiaire. En effet, au repos le distributeur envoie la pression dans les deux chambres du vérins ce qui bloque complètement la tige du vérin dans sa position.

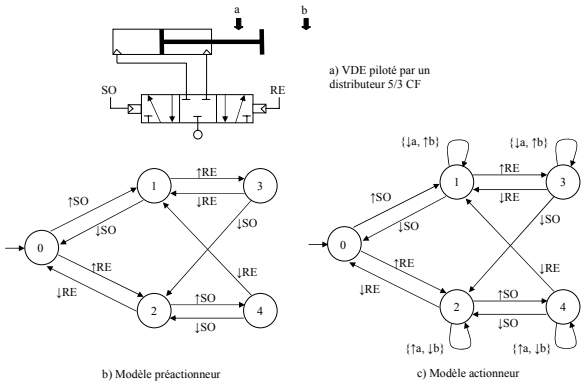


Figure 16 : VDE piloté par distributeur 5/3 CF

Le modèle préactionneur (Figure 16b) est semblable à celui du VDE piloté par un distributeur 5/3 CO. Le modèle actionneur quant à lui évolue différemment à l'état initial (Figure 16c). A l'activation de l'ordre *SO* (état 1) et sa désactivation (retour en 0), le distributeur retourne en position intermédiaire. Le vérin va alors se retrouver bloquer dans sa position par la pression présente dans les deux chambres du vérin. L'état 0 ne possède donc aucune réaction au niveau des détecteurs jusqu'à l'activation d'un des deux ordres.

6. Moteur à un sens de rotation

Les moteurs avec un seul sens de rotation sont utilisés très souvent comme organe d'un tapis de transfert de pièce. Leur modélisation dépend du déplacement de la pièce sur le tapis et notamment du lieu de chargement et de déchargement de celle-ci. Si l'on considère l'exemple de la Figure 17a où le détecteur se trouve entre le poste de chargement et le poste de déchargement, alors le modèle de préactionneur ne peut qu'activer ou désactiver l'ordre *D* du moteur (Figure 17b). Le modèle actionneur ne possède dans ce cas que deux états où seul l'état 1 de la Figure 17c autorise tous les événements liés aux détecteurs.

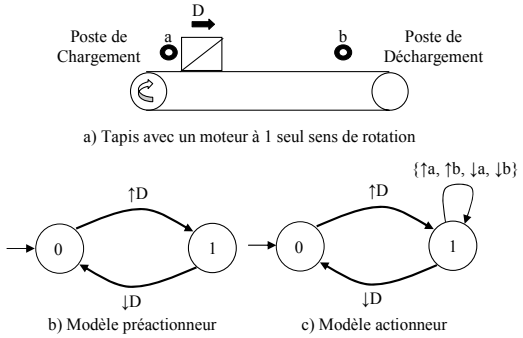


Figure 17 : Modélisation d'un moteur à un sens de rotation

7. Moteur à deux sens de rotation

La modélisation d'un contacteur d'un moteur à deux sens de rotation, pour le déplacement de chariot par exemple (Figure 18), aboutit au même modèle de préactionneur et d'actionneur

d’un VDE piloté par un distributeur 5/3 Centre Ouvert de la Figure 15. En effet, après l’activation et la désactivation d’un ordre, le moteur s’arrête à sa position actuelle. Soit l’on considère qu’il peut se déplacer manuellement, alors tous les événements sur les détecteurs peuvent apparaître à l’état de repos, soit le moteur bloque toutes évolutions possibles et n’évolue pas tant qu’il n’y a pas de nouvel ordre, ce qui entraîne un modèle sans réaction possible sur l’état de repos 0.

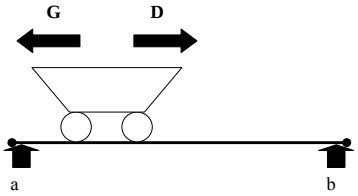


Figure 18 : Chariot avec moteur à deux sens de rotation

Dans nos travaux, nous utiliserons toujours un modèle pratique définit dans cette bibliothèque d’EPO.

RESUME

Mots-clés : Diagnostic, Systèmes à Evénements Discrets, approche décentralisée, systèmes manufacturiers, modélisation.

Le diagnostic des défaillances des systèmes industriels est à l'origine de nombreux travaux depuis ces dernières années. Il est défini comme l'opération permettant de détecter et de localiser un défaut. La détection de défauts consiste à rendre une décision sur l'état du système qu'il soit en fonctionnement normal ou défaillant. Cette opération est ensuite suivie d'une étape de localisation du défaut afin d'identifier ses causes et son origine.

Ce mémoire de thèse présente une approche décentralisée avec coordinateur pour le diagnostic des Systèmes à Evénements Discrets (SED) et plus particulièrement pour les systèmes manufacturiers composés de capteurs et d'actionneurs discrets. Cette approche considère la Partie Opérative (PO) comme un ensemble d'éléments composé d'un actionneur et d'un ensemble de capteurs. La construction des diagnostiqueurs s'appuie sur une modélisation modulaire des éléments de la Partie Opérative, d'un modèle des spécifications de la Partie Commande (PC) et d'une information temporelle liée à la réactivité des actionneurs. Chaque diagnostiqueur représente un observateur de l'état du système affecté d'une étiquette de décision, cette décision étant le résultat de l'observation des événements, des conditions sur les états du système et/ou sur le temps de retard entre événements.

Afin de définir la capacité de l'ensemble des diagnostiqueurs locaux à diagnostiquer un ensemble de défauts dans un délai fini, une notion de codiagnosticabilité a été établie. Dans cette thèse, cette notion tient compte de la modélisation des défauts à base d'événements, à base d'états et à base d'informations temporelles. Elle détermine ainsi l'ensemble des défauts que la structure décentralisée peut diagnostiquer.

L'ensemble des décisions locales doit être ensuite agrégé afin d'obtenir une décision globale sur l'état du système. Cette fusion est réalisée par un coordinateur construit à partir d'un ensemble de règles permettant de résoudre les différents problèmes d'indécision et d'ambiguïté entre les diagnostiqueurs locaux. Ce coordinateur permet d'obtenir des performances de diagnostic équivalentes à celles d'un diagnostiqueur centralisé. Deux exemples d'applications manufacturières illustrent l'efficacité et l'intérêt de la structure décentralisée en terme d'explosion combinatoire. Un simulateur basé sur *Stateflow* de Matlab permet de tester et valider l'approche proposée.

Keywords : Diagnosis, Discrete Event Systems, decentralized approach, manufacturing systems, modelling.

Fault diagnosis of industrial systems is a subject that has received a great attention in the past few decades. It is defined as the process of detecting and isolating faults. Fault detection leads to a binary decision that either the system is working under normal conditions or an abnormality in its behaviour has occurred. Fault detection is followed by fault isolation, which determines the fault type, location and causes.

This thesis presents a decentralized approach to realize the diagnosis of Discrete Event Systems (DES), particularly manufacturing systems with discrete sensors and actuators. This approach considers the plant as a set of elements. Each plant element is composed of an actuator and a set of sensors. The construction of the local diagnosers is based on a modular modelling of the plant elements, of the controller specifications and the temporal information about the actuators reactivity. It is a special case of an observer that carries fault information by means of labels attached to states. Diagnosers base their decisions on the sequences of observed events, on the current behavior and/or on the time delays between these events.

In order to verify that the set of local diagnosers are capable to diagnose a set of faults within a bounded delay, a notion of codiagnosability must be defined. In this thesis, a timed-event-state-based diagnosability notion is defined in order to verify the diagnosability property of the set of local diagnosers.

All local diagnosis decisions must be merged in order to obtain one global diagnosis decision. This fusion can be realized by a coordinator based on a set of rules. The goal of this coordinator is to solve the problem of decision conflict and/or ambiguity among local diagnosers in order to obtain a diagnosis performance equivalent to the one of the centralized diagnoser. Two examples of manufacturing applications are used to illustrate and to show the advantages and the interest of this approach. A simulation tool based on *Stateflow* of Matlab is constructed in order to test and validate the proposed approach on application examples.