Commutations sûres de mode pour les systèmes à événements discrets

Gregory Faraut

N° d'ordre: 2010-ISAL-0108 Année 2010

THÈSE

présentée devant

l'Institut National des Sciences Appliquées de Lyon

pour obtenir le grade de

Docteur

par

Gregory Faraut

Équipe d'accueil : Équipe EASY - Laboratoire Ampère
École Doctorale : Électronique, Électrotechnique, Automatique (EEA)
Spécialité : Automatique

Commutations sûres de mode pour les systèmes à événements discrets

---

soutenue le 7 décembre 2010 devant le jury composé de:

Rapporteurs :	Étienne Craye	Professeur des Universités LAGIS - École Centrale de Lille
	Olivier H. Roux	Professeur des Universités IRCCyN - École Centrale de Nantes
Examinateurs :	Charles André	Professeur des Universités, I3S / INRIA - Université de Nice Sophia Antipolis
	Nidhal Rezg	Professeur des Universités LGIPM / ISGMP - Université Paul Verlaine - Metz
Encadrants :	Éric Niel	Professeur des Universités AMPÈRE - INSA de Lyon
	Laurent Piétrac	Maître de Conférence AMPÈRE - INSA de Lyon

INSA Direction de la Recherche - Ecoles Doctorales – Quadriennal 2007-2010

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://sakura.cpe.fr/ED206** M. Jean Marc LANCELIN Insa : R. GOURDON	M. Jean Marc LANCELIN Université Claude Bernard Lyon 1 Bât CPE 43 bd du 11 novembre 1918 69622 VILLEURBANNE Cedex Tél : 04.72.43 13 95 Fax : lancelin@hikari.cpe.fr
E.E.A.	**ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE** **http://www.insa-lyon.fr/eea** M. Alain NICOLAS Insa : C. PLOSSU ede2a@insa-lyon.fr Secrétariat : M. LABOUNE AM. 64.43 – Fax : 64.54	M. Alain NICOLAS Ecole Centrale de Lyon Bâtiment H9 36 avenue Guy de Collongue 69134 ECULLY Tél : 04.72.18 60 97 Fax : 04 78 43 37 17 eea@ec-lyon.fr Secrétariat : M.C. HAVGOUDOUKIAN
E2M2	**EVOLUTION, ECOSYSTEME, MICROBIOLOGIE, MODELISATION** **http://biomserv.univ-lyon1.fr/E2M2** M. Jean-Pierre FLANDROIS Insa : H. CHARLES	M. Jean-Pierre FLANDROIS CNRS UMR 5558 Université Claude Bernard Lyon 1 Bât G. Mendel 43 bd du 11 novembre 1918 69622 VILLEURBANNE Cédex Tél : 04.26 23 59 50 Fax 04 26 23 59 49 06 07 53 89 13 e2m2@biomserv.univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTE** Sec : Safia Boudjema M. Didier REVEL Insa : M. LAGARDE	M. Didier REVEL Hôpital Cardiologique de Lyon Bâtiment Central 28 Avenue Doyen Lépine 69500 BRON Tél : 04.72.68 49 09 Fax :04 72 35 49 16 Didier.revel@creatis.uni-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHEMATIQUES** http://infomaths.univ-lyon1.fr M. Alain MILLE	M. Alain MILLE Université Claude Bernard Lyon 1 LIRIS - INFOMATHS Bâtiment Nautibus 43 bd du 11 novembre 1918 69622 VILLEURBANNE Cedex Tél : 04.72. 44 82 94 Fax 04 72 43 13 10 infomaths@bat710.univ-lyon1.fr - alain.mille@liris.cnrs.fr
Matériaux	**MATERIAUX DE LYON** M. Jean Marc PELLETIER Secrétariat : C. BERNAVON 83.85	M. Jean Marc PELLETIER INSA de Lyon MATEIS Bâtiment Blaise Pascal 7 avenue Jean Capelle 69621 VILLEURBANNE Cédex Tél : 04.72.43 83 18 Fax 04 72 43 85 28 Jean-marc.Pelletier@insa-lyon.fr
MEGA	**MECANIQUE, ENERGETIQUE, GENIE CIVIL, ACOUSTIQUE** M. Jean Louis GUYADER Secrétariat : M. LABOUNE PM : 71.70 –Fax : 87.12	M. Jean Louis GUYADER INSA de Lyon Laboratoire de Vibrations et Acoustique Bâtiment Antoine de Saint Exupéry 25 bis avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél :04.72.18.71.70 Fax : 04 72 43 72 37 mega@lva.insa-lyon.fr
ScSo	**ScSo*** M. OBADIA Lionel Insa : J.Y. TOUSSAINT	M. OBADIA Lionel Université Lyon 2 86 rue Pasteur 69365 LYON Cedex 07 Tél : 04.78.77.23.88 Fax : 04.37.28.04.48 Lionel.Obadia@univ-lyon2.fr

*ScSo : Histoire, Geographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Remerciements

Le travail présenté dans ce mémoire a débuté au sein du Laboratoire d'Automatique Industrielle (LAI) de l'INSA de Lyon qui est devenu par la suite le laboratoire AMPÈRE. À ce titre, je tiens à remercier pour leur accueil M. Tanneguy Redarce, Professeur des Universités et directeur à mon arrivée, et M. Laurent Nicolas, Directeur de Recherche CNRS et actuel directeur du laboratoire. Je tiens naturellement à remercier mon directeur de thèse, M. Eric Niel, Professeur des Universités, pour la confiance qu'il m'a accordé et pour les nombreuses discussions que nous avons eu sur mes travaux et la recherche en générale. Mes remerciements vont bien évidemment aussi à M. Laurent Piétrac, Maître de Conférences, pour m'avoir encadré, parfois supporté, au jour le jour, pour m'avoir clairement fait monter en compétences, pour la rigueur qu'il a inséré dans mes méthodes de travail, et enfin, pour la remise en question perpétuelle des travaux. Ainsi, je leur exprime ma plus profonde gratitude pour l'énergie qu'ils ont dépensé pour m'aider à finaliser cette thèse, du début à la fin, et bien que je ne sois pertinemment pas objectif, je pense pouvoir dire que je n'aurai pu avoir meilleurs encadrants. J'aimerai également exprimer ma gratitude envers chacun des membres du jury de thèse ; A M. Étienne Craye, Professeur des Universités et directeur de l'École Centrale de Lille, et M. Olivier H. Roux, Professeur des Universités, pour avoir accepté d'être les rapporteurs de mes travaux. Les échanges que nous avons eu ont toujours été agréables et forts instructifs. Á M. Nidhal Rezg, Professeur des Universités, pour avoir accepté d'examiner mon travail et de présider le jury de thèse, et enfin à M. Charles André, Professeur des Universités, pour l'examen de mes travaux et ses nombreuses remarques sur le manuscrit. La soutenance de thèse fut intense mais l'échange qui en a résulté a été un des moments les plus enrichissants que j'ai pu vivre durant ces dernières années. Sans vouloir forcément revivre ce moment, je les remercie encore une fois grandement pour avoir accepté d'y participer. Parce qu'une thèse se prépare dans un laboratoire, j'aimerai remercier les membres du laboratoire AMPÈRE, et plus particulièrement ceux que j'ai pu croiser dans ce long couloir, qui a amené tant de discussions passionnantes : Daniel, Damien, Sylvie, Éric, Xavier, Minh Tu, Mickaël, Wilfried, Jean-Pierre, Émil et Richard. Merci également à Maguy, pour sa compétence administrative impressionnante et sa boite à bonbons jamais vide, ainsi qu'à Christophe pour son aide technique lors de besoins particuliers, notamment d'accessibilité au

réseau. Il est également important de ne pas se sentir seul dans certaines situations, de ce fait, je voudrais remercier les autres doctorants que j'ai pu croiser durant ces années : Audrey, pour ces mots d'encouragement quotidiens, Sajeh, pour nos longues discussions remplies de rires, Salam, pour sa gentillesse sans égale, ainsi que Fred, Gerardo, Juan, Sylvain, Quyen, Lilia, Mariem, Mingming, Anouar et enfin Yves (qui n'est pas doctorant mais qui n'est pas passé loin !). Durant ma thèse, j'ai eu la chance de pouvoir transmettre une partie de mes connaissances à de jeunes étudiants motivés et pleins d'entrain pour l'acquisition de nouveaux savoirs. Ceci m'a permis de rencontrer une formidable équipe qui a beaucoup contribué à ma façon de voir le métier d'enseignant. Je souhaite donc remercie chaleureusement les enseignants du département de Génie industriel de l'Insa de Lyon, à savoir : Corinne (évidemment !), Laurent, Samir, Fred, Julien, Stéphane, Hélène, Christine, Françoise, Arnaud, Madeth, Audrey, Alain, Jean-Pierre, Patrick, Thierry, Valérie, Pascale, Lorraine et enfin les autres personnels de ce département, Nathalie, Christine et Olivier. J'ai eu l'opportunité de réaliser un séjour dans un laboratoire étranger. Ceci grâce à une bourse Explora'Doc de la région Rhône-Alpes. À cet égard, je tiens à remercier la région pour avoir financé le séjour. Ce séjour s'est déroulé à l'Université du Michigan, au département EECS, et plus particulièrement dans l'équipe UMDES pour University of Michigan - Discrete Event Systems. Je remercie très chaleureusement le professeur Stéphane Lafortune pour son accueil au sein de l'équipe, nos réunions hebdomadaires où il n'a eu de cesse de me débloquer et m'aider dans les chemins à explorer. Je lui dois vraiment une partie de ces travaux. Cependant, l'hiver au Michigan est rude (quand il fait zéreau, c'est qu'il fait bo !), et les relations humaines et amicales prennent alors toute leur importance. Pour cela, je tiens à remercier : Nicolas, premier français rencontré là-bas et avec qui les soirées burgers-bières à discuter de la vie américaine me rendent nostalgique, Arnau et Dalal, pour leur bonne humeur et leur sourire permanent, Hongwei, pour sa sympathie et nos échanges intéressants sur la Chine (et plus particulièrement sa gastronomie), Manu (pour qui la recherche, c'est sympa, mais un jour faut quand même vraiment travailler), Oliver (le seul ami anglais que j'ai jamais eu), Ru (beaucoup de chose à dire, mais je me contenterai de dire qu'elle fait magnifiquement bien la cuisine), et Dan (pour nos soirées en coop sur Resident Evil 5, ce n'était vraiment pas sérieux). Vous m'avez réchauffé le cœur dans ce territoire frigorifique. D'un point de vue plus personnel, je voudrais remercier ma famille qui, loin de comprendre les raisons qui m'ont poussé à faire une thèse, ont toujours respecté mon choix et encouragé à aller jusqu'au bout. Il est certain que sans eux je n'aurai pu aller si loin et je les remercie du fond du cœur. Un avant-dernier petit mot va à ma seconde famille qu'est la Croix-Rouge Française. Je la remercie d'avoir toujours trouvé à m'occuper, souvent au-delà du raisonnable, mais surtout d'avoir contribué à équilibrer ma vie entre travail et social. Ainsi, je tiens à remercier ceux qui passent trop de temps au 15, av. Jean Mermoz : Rémicounet, Clairette, Schtroumpfette, Caro, Christelle, Zabou, LN, Sarah, Sarah DK, Léa, Lapsy, Fred, Laure, Karine B., Karine R., Cécile, Anne, Armand, Arnaud, Camille, Pierre-Luc, Quentin, Vincent, Loïc, Joël, Rémi, et j'en oublie forcément et malheureusement,

qu'ils me pardonnent. Enfin, je ne peux oublier les amis de longue date, de ma ville d'origine, qui m'ont fait comprendre qu'ils seraient toujours là malgré la distance. Alors un énorme merci à Marilou et Goubs (avec Grenouille et Crocodile), à ma Grosse (Steph), à Jul et Isa (et la petite Olivia), à Corda et Xiou (promis je viens vous rendre visite bientôt), à Yo et Émilie (bonne vie à vous deux :P), à Alexandra et Fred (couple creuf), à Laulau (Courage pour la fin de ta thèse), Steph et Mag (et leur p'tit bout'chou), à Chacha et Lize Les colocs de toujours, à Mon Amour et Alain (et leur dernier PC), à Willounet (et notre seconde maison, comprenant Roberto), à Linda (courage pour le retour dans le sud), et pour finir à mes 2 acolytes de naissances que sont Guigui et Cedr'X.

Je ne sais pas ce que je serai devenu sans vous, mais je sais ce que je suis devenu grâce à vous tous.

Greg.

Table des matières

Préambule

Les systèmes à Événements discrets (SED) recouvrent une grande variété de systèmes tels que les systèmes embarqués, les systèmes de production, les protocoles de communication, etc [Cas07]. En se focalisant sur la maîtrise de leur comportement, une des notions couramment utilisée pour la conception de la commande de ces systèmes est la notion de mode, qu'elle corresponde à un changement de type de pilotage (manuel, semi-automatique, automatique), à un changement d'objectif (mode nominal, mode de test) ou à un changement de configuration du système piloté (train d'atterrissage rentré/sorti, actionneur en panne...). Si une telle appréhension par mode (approche dite modale) se prête par nature à la considération des exigences fonctionnelles et sécuritaires, elle diminue de fait considérablement la complexité des modèles manipulés par la décomposition. Ces modèles ne s'intéressent en effet qu'à une période de fonctionnement et à un ensemble de composants plus réduits. En outre, dans un mode particulier, le fonctionnement du système ne nécessite pas toujours tous ses composants. Par ailleurs, cela évite au concepteur d'étudier tous les problèmes en même temps.

À ce jour, très peu de méthodes ont été spécifiquement conçues pour appréhender la gestion des modes. Les plus usuelles reposent sur les approches classiques de conception, laissant au concepteur la responsabilité de spécifier un comportement qui sera ensuite implanté dans la commande. Il s'avère alors nécessaire, au moins sur les systèmes critiques, de vérifier et de valider *a posteriori* ce comportement. Dans le cadre de la gestion des modes, l'exploitation de ce type d'approche nécessiterait la vérification/validation des comportements internes à chacun des modes puis ceux liés aux changements de mode. En effet, la problématique majeure en gestion des modes est celle des commutations (i.e. des passages de mode en mode).

À l'opposé de ces méthodes, la théorie du contrôle par supervision (SCT pour *Supervisory Control Theory*) permet de prendre en compte *a priori* les propriétés attendues du système pour synthétiser l'ensemble des comportements admissibles [Ram87]. Le concepteur garde bien sûr la responsabilité de la spécification de ces propriétés, comme dans les approches classiques, mais une erreur dans ces spécifications peut être détectée plus tôt.

Les travaux de Nourelfath [Nou97] ont été les premiers à proposer d'utiliser la SCT pour intégrer à l'étude du mode nominal la prise en compte des comportements admissibles du sys-

tème après une panne d'un de ses composants. Ces travaux ont permis d'introduire, pour résoudre le problème lié à l'utilisation de plusieurs modèles, les notions d'état inactif et de suivi de trajectoire. Cependant, ils ne permettaient d'étudier que deux modes de fonctionnement : le fonctionnement nominal et le fonctionnement dégradé. Cette limitation à deux modes était due aux choix de conception dans la démarche proposée. L'extension supprimant cette limitation a été proposée dans les travaux de Kamach [Kam04]. Ces travaux ont reconstitué un contexte plus formel de ces notions d'états inactifs et de suivi de trajectoire, toujours dans le cadre de la SCT. Ils ont également démontré l'unicité d'un superviseur par mode. Néanmoins, la démarche proposée nécessite que le concepteur spécifie toujours manuellement et par supposition les trajectoires de commutation entre modes du procédé sous contrôle. Ceci est le point de départ de nos travaux pour lesquels nous proposons une démarche où le concepteur exprimera les propriétés fonctionnelles et sécuritaires de la commande à respecter lors des commutations pour obtenir par construction les comportements admissibles.

Ce mémoire est composé de quatre chapitres, regroupés en deux parties. La première partie comporte les deux premiers chapitres. Le premier chapitre est consacré à la présentation des bases théoriques de la SCT et aux approches décompositionnelles. Le deuxième chapitre est dédié à un état de l'art des travaux portant sur la modélisation des modes. Pour chacun des travaux, nous présenterons le point de vue adopté ainsi que les formalismes et méthodes utilisés. Une synthèse de ces travaux sera présentée pour identifier les idées communes et les choix les plus pertinents. C'est à l'issu de ce chapitre que nous exposerons les objectifs de notre proposition.

La seconde partie, dédiée à notre proposition, comporte les deux chapitres suivants. Le chapitre 3 présente la démarche proposée, décomposée en plusieurs étapes. La première est la modélisation. C'est une étape de représentation par un formalisme état-transition de la dynamique libre des composants. Ce formalisme de représentation est également utilisé pour représenter les spécifications fonctionnelles et sécuritaires sur ces composants. La deuxième étape suivante est celle de l'approche modale, où le comportement interne à chaque mode est élaboré en adéquation avec la notion de mode adoptée dans ces travaux. L'étape suivante se décompose en deux activités qui, pris ensemble, s'intéressent à l'étude et à la résolution de l'admissibilité des commutations de modes, expressions des trajectoires de commutation autorisées. Enfin, la dernière étape concerne la réduction de la taille des modèles finaux dans le but d'obtenir un modèle par mode. Le dernier chapitre valide nos propositions sur un exemple proposé dans la littérature.

Nous concluons enfin ce mémoire en faisant un bilan de notre apport et en évoquant des perspectives de suite à notre travail.

1

Bases théoriques

Sommaire

1.1 Introduction

Nous allons montrer dans ce premier chapitre le contexte dans lequel nous nous trouvons, à savoir les systèmes à événements discrets. Nous présentons également la théorie des langages et automates et la théorie du contrôle par supervision afin de définir le cadre de base sur lequel reposeront nos propositions.

1.1.1 Les systèmes à événements discrets

Un système est un ensemble d'éléments, élémentaires ou pas, qui interagissent entre eux, afin de réaliser une fonction [Rad97]. Les systèmes à événements discrets (SED) sont une classe particulière de systèmes : ce sont des systèmes dynamiques qui ont un espace d'états discret et qui évoluent suivant l'occurrence d'événements [Cas07]. Un événement représente indifféremment une action (une personne appuyant sur un bouton d'urgence), un début ou une fin de tâche (fin de cycle de nettoyage d'une machine à laver) ou l'arrivée d'un problème inopiné (impossibilité de sortir le train d'atterrissage). Un événement peut être provoqué par un être humain, par une machine, par la nature ou par toute combinaison possible d'actions, de problèmes, etc. de différents acteurs. Quel qu'il soit, l'occurrence d'un événement est asynchrone, il n'est pas synchronisé à une horloge, et instantané.

De nombreux systèmes peuvent être vus comme des systèmes à événements discrets. Les systèmes incluant des guichets et des files d'attente évoluent par exemple en fonction des événements de prise d'un ticket ou d'accès au guichet. Les systèmes manufacturiers, dans lequel l'état des machines évolue en fonction des événements de production sont aussi des SED. Les systèmes informatiques ou les systèmes de communications sont d'autres exemples concrets, mais un SED peut aussi être un système plus abstrait, tel que les exemples classiques de la table des philosophes ou le voyageur de commerce.

L'étude des SED, et des systèmes en général, peut se faire en fonction d'objectifs différents. L'évaluation de performance, l'optimisation, le dimensionnement, la conception du contrôle-commande en sont quelques exemples. Quel que soit l'objectif, cette étude passe par des modèles construits grâce à des langages, dans le cadre de théories ou de méthodes. Pour l'étude de la commande des systèmes, de nombreux langages ont été proposés : le Grafcet [Zay99], les Réseaux de Petri [Pet81, Mur89], les Statecharts [Har87] font partis des plus connus. Le succès de ces langages est notamment dû à leur aspect graphique, ce qui leur permet d'être un excellent support de discussion entre les concepteurs et leurs clients.

Cependant, toutes les méthodes utilisant ces langages impliquent que le concepteur crée les modèles à partir d'un cahier des charges, la plupart du temps imprécis et incomplet, parfois contradictoire. Depuis quelques années, de nombreux travaux de recherche ont donc proposé de compléter cette étape de construction de modèles par une étape de vérification de propriétés.

Ces propriétés, toujours interprétées à partir du cahier des charges, ne peuvent être exploitées que si elles sont exprimées dans un langage dont la sémantique est mathématiquement définie. Malheureusement, dans toutes ces méthodes, les propriétés ne sont prises en compte qu'*a posteriori*, i.e. après la construction des modèles de commande.

1.1.2 Principes de la Théorie de Contrôle par Supervision

Initiée par Ramadge et Wonham [Ram87, Ram89], la théorie du contrôle par supervision propose au contraire de synthétiser le modèle de commande à partir, entre autres, des propriétés. Pour ce faire, elle repose sur la distinction entre les modèles du procédé à contrôler, des propriétés à respecter ou du comportement souhaité et de la structure de contrôle.

La figure 1.1 illustre ainsi la démarche courante d'utilisation de cette théorie, dans laquelle la première étape (1) consiste à construire d'une part le modèle du procédé, représentant les comportements possibles du système étudié, et le modèle des spécifications, représentant les propriétés attendues, d'autre part.

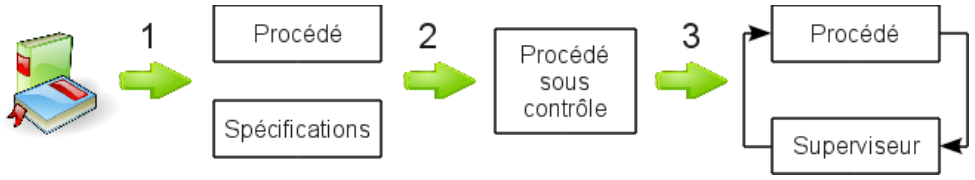

FIGURE 1.1 – Principe de conception dans la théorie du contrôle par supervision

À partir de ces deux modèles, l'étape suivante (2) consiste à obtenir mathématiquement le modèle du procédé sous contrôle dont le comportement représente ce qui est possible dans le procédé tout en respectant les contraintes imposées par les spécifications. Enfin, à l'étape (3), le superviseur, dont le rôle est de restreindre les comportements possibles du procédé, peut être synthétisé. Le comportement en parallèle du procédé et du superviseur est alors le même que celui du procédé sous contrôle.

L'avantage de cette séparation entre modèles est double. Le premier est de ne plus mélanger plusieurs concepts dans un seul modèle construit par le concepteur. Cela revient à séparer les points de vue, comportements possibles et spécifications, pour mieux exprimer chacun. Le second est la synthèse des comportements admissibles. En effet le modèle du procédé sous contrôle (ou du procédé+superviseur) représente l'ensemble des comportements possibles. Le concepteur pourra donc choisir le comportement qu'il souhaite implanter, y compris parmi des comportements qu'il n'avait pas imaginé.

Un des concepts fondateurs de la théorie du contrôle par supervision est la notion de contrôlabilité. Associée au concept d'événement, cette propriété permet d'autoriser le superviseur à interdire ou pas un événement, et donc un comportement possible du procédé. Le langage d'origine de la théorie du contrôle par supervision est l'automate à états [Sip05], langage dans lequel

la contrôlabilité d'un événement peut être facilement intégrée, y compris graphiquement. Nos travaux étant basés sur cette théorie et sur l'utilisation des automates à états, la suite de ce chapitre sera consacrée à leur présentation.

Remarque 1
D'autres propriétés telles que l'observabilité et la diagnosticabilité on été plus récemment introduites favorisant l'étude de champs complémentaires à la commande comme l'identification et le diagnostic. Cependant, ces propriétés ne seront pas exploitées par la suite. ♦

1.2 Théorie des langages et Automates

L'origine de la théorie des langages vient de travaux linguistiques cherchant à caractériser les langages selon des références sémantiques définies. L'intérêt est d'identifier les règles élémentaires de construction d'un langage afin d'être capable par le suite d'étendre ces règles et de manipuler mathématiquement ces langages [Cho59].

1.2.1 Symboles, alphabet, mots et langage

Dans la théorie des langages, l'élément atomique est le *symbole*. Il n'existe pas de règle définissant ce que doit être un symbole. Les éléments graphiques constituant un symbole sont généralement un ou plusieurs des éléments suivants : lettres, chiffres, tout autre caractère (dièse, point d'exclamation, etc), dessins (hiéroglyphes).

Un *alphabet*, souvent désigné par Σ, est un ensemble fini de symboles. C'est à partir de cet ensemble, et des symboles qui le constituent, qu'il est possible de construire des mots.

Un *mot*, noté s, sur l'alphabet Σ, est une suite finie, fabriquée par *concaténation*, $s_1.s_2 \dots s_n$ de symboles inclus dans Σ. Par soucis de simplification, l'écriture $s_1 s_2$ est considéré comme équivalente à $s_1.s_2$. Le mot vide est un mot particulier et est communément noté ε. L'ensemble de tous les mots qu'il est possible de créer avec les symboles de l'alphabet Σ est appelé Σ^* (ε est inclus dans Σ^*).

La concaténation de deux mots u et v tels que $u = u_1 u_2 \dots u_n$ et $v = v_1 v_2 \dots v_m$ est le mot noté $u.v$, ou uv égal à $u_1 u_2 \dots u_n v_1 v_2 \dots v_m$ obtenu par simple juxtaposition. Cette opération est associative et non commutative. Lorsqu'un mot s constitué par concaténation de trois autres mots t, u, v tel que $s = tuv$, la terminologie est la suivante : t est appelé le *préfixe* de s, u est appelé le *sous-mot* de s, et v est appelé le *suffixe* de s.

Un *langage*, noté L, est un ensemble de mots constitués avec les symboles de l'alphabet Σ. Par conséquent, un langage L est inclus ou égal à l'ensemble Σ^* constitué par tous les mots réalisables sur l'alphabet Σ.

1.2.2 Opérations sur les langages

Les langages étant considérés mathématiquement comme des ensembles, il est possible de leur appliquer tous les opérateurs issus de la théorie des ensembles (union, intersection...). Il est aussi possible de leur appliquer les opérations présentés dans cette section.

Définition 1 (Concaténation)
Pour deux langages L_a et L_b tels que $L_a, L_b \subseteq \Sigma^*$:

$$L_a L_b = \{s \in \Sigma^* | (s = s_a s_b) et (s_a \in L_a) et (s_b \in L_b)\} \quad \blacklozenge$$

Autrement dit, tous les mots de $L_a L_b$ peuvent être écrits comme un mot contenu dans L_a immédiatement suivi d'un mot contenu dans L_b.

Définition 2 (Préfixe-clos)
Pour un langage L tel que $L \subseteq \Sigma^*$:

$$\overline{L} = \{s \in \Sigma^* | (\exists t \in \Sigma^*)[st \in L]\} \quad \blacklozenge$$

Le préfixe-clos de L, noté $\overline{L}$, est construit avec tous les préfixes de tous les mots de L.

Définition 3 (Étoile)
Pour un langage L tel que $L \subseteq \Sigma^*$:

$$L^0 = \{\varepsilon\},\ L^{i+1} = LL^i,\ L^* = \bigcup_{i \geq 0} L^i \quad \blacklozenge$$

L'opérateur étoile (*), appliqué à un langage, donne l'ensemble résultant de l'union du langage L concaténé avec lui-même à chaque occurrence.

Définition 4 (Projection)
Considérant deux alphabets Σ_1 et Σ_2 tels que $\Sigma_2 \subseteq \Sigma_1$. La fonction de projection $P_{1,2}$ est définie telle que :

$$P_{1,2} : \Sigma_1^* \rightarrow \Sigma_2^* \quad où :$$
$$P_{1,2}(\varepsilon) = \varepsilon$$
$$P_{1,2}(\sigma) = \begin{cases} \sigma \ \ si \ \ \sigma \in \Sigma_2 \\ \varepsilon \ \ si \ \ \sigma \in \Sigma_1 \backslash \Sigma_2 \end{cases}$$
$$P_{1,2}(s\sigma) = P_{1,2}(s) P_{1,2}(\sigma) \ avec \ s \in \Sigma_1^*, \sigma \in \Sigma_1$$

Appliquée au langage, en considérant un langage L tel que $L \in \Sigma_1^*$, la projection d'un langage est définie par :

$$P_{1,2}(L) = \{t \in \Sigma_2^* | (\exists s \in L)[P_{1,2}(s) = t]\} \quad \blacklozenge$$

Définition 5 (Projection Inverse)
Considérant deux alphabets Σ_1 et Σ_2 tels que $\Sigma_2 \subseteq \Sigma_1$. La fonction de projection inverse $P_{2,1}^{-1}$ est définie telle que :

$$P_{2,1}^{-1} : \Sigma_2^* \rightarrow 2^{\Sigma_1^*} \qquad avec \qquad P_{2,1}^{-1}(t) = \{s \in \Sigma_1^* | P_{1,2}(s) = t\}$$

Appliquée aux langages, et en considérant un langage L tel que $L \in \Sigma_2^*$, la projection inverse d'un langage est définie par :

$$P_{2,1}^{-1}(L) = \{s \in \Sigma_1^* | (\exists t \in L)[P_{1,2}(s) = t]\} \quad \blacklozenge$$

Ces définitions constituent une part non négligeable des fonctions nécessaire à la manipulation des langages. Celles-ci peuvent être retrouvées dans différents ouvrages tels que [Cas07, Won09].

1.2.3 Automate et Langage régulier

Un automate à états finis est un des modèles le plus simple qui existe permettant d'effectuer des calculs [Sip05]. Ce modèle est suffisant pour représenter un grand nombre de systèmes différents. Formellement, le modèle d'un automate à état finis est défini comme suit :

Définition 6 (Automate)
Un automate G est défini par un quintuplet $G = (Q, \Sigma, \delta, q_0, Q_m)$ tel que :

- *Q est un ensemble fini contenant tous les états de G ;*
- *Σ est l'ensemble des événements, aussi appelé alphabet, de G ;*
- *δ est la fonction de transition, définie par $\delta : Q \times \Sigma \rightarrow Q$. $\delta(x, \sigma) = y$ représente une transition portant l'événement σ qui mène de l'état x à l'état y ;*
- *q_0 est l'état initial de l'automate G, tel que $q_0 \in Q$;*
- *Q_m est l'ensemble des états marqués de G, tel que $Q_m \subseteq Q$.* $\blacklozenge$

Un exemple d'automate à état finis est illustré figure 1.2. Cet automate représente le comportement d'une machine et possède trois états : (*A*)rrêt, (*M*)arche et (*P*)anne. L'état *A* est l'état initial de notre automate. il est symbolisé graphiquement par une flèche entrante dans cet état, mais ne sortant d'aucun autre état. Cet état *A* est de plus considéré comme le seul état marqué de la machine ($Q_m = \{A\}$). Graphiquement, un état marqué est représenté par une flèche sortante (ne menant pas dans un autre état) ou par un deuxième cercle entourant l'état. Ici, l'état *A* est

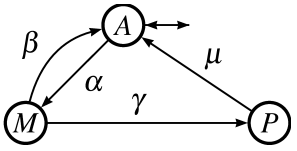

FIGURE 1.2 – Exemple d'automate

à la fois état initial et état marqué, d'où la double flèche. Un état marqué symbolise une fin de tâche ou un objectif à atteindre : la machine doit être à l'arrêt en fin de production.

De cet état A sort une flèche portant l'événement α et entrant dans l'état M. Cette flèche représente la transition $\delta(A,\alpha) = M$. L'ensemble des flèches représente donc la fonction de transition $\{((A,\alpha),M),((M,\beta),A),((M,\gamma),P),((P,\mu),A)\}$.

La particularité d'un automate est de représenter, ou générer, un langage. Dans ce langage est inclus le langage marqué qui contient tous les mots menant à un état marqué. Formellement, cela conduit aux définitions suivantes :

Définition 7 (Langage généré et marqué)
Le langage généré par $G = (Q,\Sigma,\delta,q_0,Q_m)$ *est défini par :*

$$L(G) = \{s \in \Sigma^* | \delta(q_0,s) \textit{est défini}\}$$

Le langage marqué par G est défini par :

$$L_m(G) = \{s \in L(G) | \delta(q_0,s) \in Q_m\}$$ ♦

Ces deux dernières définitions supposent bien entendu que la fonction de transition soit étendue telle que : $\delta : Q \times \Sigma^* \to Q$. Une propriété intéressante du langage généré par un automate est qu'il est, par construction, préfixe-clos : $L(G) = \overline{L(G)}$.

Les langages rationnels ont été définis dans la hiérarchie de Chomsky [Cho59] où ils occupent le troisième niveau. Leur particularité est d'être représentable par un automate à états fini. Le théorème suivant énonce la relation entre langage rationnel et automate. Sa démonstration fut énoncée par Kleene et peut être trouvée dans [Car07].

Théorème 1 (Langage Rationnel)
Un langage L est dit rationnel si et seulement si il existe un automate à états finis G tel que $L = L_m(G)$. ♦

1.2.4 Opérations sur les automates

Dans un automate G, un état *accessible* est un état atteignable à partir de l'état initial q_0, c'est-à-dire un état q pour lequel un mot $s \in \Sigma^*$ tel que $\delta(q_0, s) = q$ existe. Il est donc possible d'extraire de l'automate G l'automate $Ac(G)$ ne contenant que les états accessibles de G :

Définition 8 (Partie accessible d'un automate)
Considérant un automate G tel que $G = (Q, \Sigma, \delta, q_0, Q_m)$, sa partie accessible $Ac(G)$ est définie telle que $Ac(G) = (Q_{ac}, \Sigma, \delta_{ac}, q_0, Q_{m,ac})$ où :

- $Q_{ac} = \{q \in Q | (\exists s \in \Sigma^*)[\delta(q_o, s) = q]\}$
- $Q_{m,ac} = Q_m \cap Q_{ac}$
- $\delta_{ac} = \delta|_{Q_{ac} \times \Sigma \to Q_{ac}}$ *(la notation « | » signifie « restreint à »)* ♦

Cette opération ne change pas le langage $L(G)$ et le langage marqué $L_m(G)$ car les mots de ces langages sont générés à partir de l'état initial. Or, avec cette opération tous les états non accessibles depuis l'état initial sont supprimés. Seules la taille de l'automate et la fonction de transfert sont réduites.

Les états q co-accessibles sont quant à eux tous les états de G permettant d'atteindre un état marqué, c'est-à-dire tous les états q pour lesquels il existe un mot $s \in \Sigma^*$ tel que $\delta(q, s) \in Q_m$ existe. Il est donc aussi possible d'extraire de G l'automate $CoAc(G)$ ne contenant que les états co-accessibles de G :

Définition 9 (Partie co-accessible d'un automate)
Considérant un automate G tel que $G = (Q, \Sigma, \delta, q_0, Q_m)$, sa partie co-accessible $CoAc(G)$ est définie telle que $CoAc(G) = (Q_{coac}, \Sigma, \delta_{coac}, q_{0,coac}, Q_m)$ où :

- $Q_{coac} = \{q \in Q | (\exists s \in \Sigma^*)[\delta(q, s) \in Q_m]\}$
- $q_{0,coac} = \begin{cases} q_0 & si\ \ q_0 \in Q_{coac} \\ non\ défini & sinon \end{cases}$
- $\delta_{coac} = \delta|_{Q_{coac} \times \Sigma \to Q_{coac}}$ ♦

Cette opération sur l'automate G peut tronquer son langage généré en supprimant certains état accessibles depuis l'état initial. Cependant cette opération n'affecte pas le langage marqué qui reste identique. Ainsi, dans le cas où $G = CoAc(G)$, nous avons alors l'égalité suivante : $L(G) = \overline{L_m(G)}$

Un automate dans lequel tous les états sont à la fois accessibles et co-accessibles est appelé un automate émondé. La recherche de cet automate revient à définir l'opération suivante :

Définition 10 (Partie émondée d'un automate)

$$Em(G) = Ac[CoAc(G)] = CoAc[Ac(G)]$$

Une autre opération très utilisée est la composition parallèle d'automates qui permet de représenter dans un seul et unique automate le comportement parallélisé des automates ayant servi à sa construction. Formellement, la composition parallèle est défini comme suit :

Définition 11 (Composition Parallèle)
Considérant deux automates $G_1 = (Q_1, \Sigma_1, \delta_1, q_{0,1}, Q_{m,1})$ *et* $G_2 = (Q_2, \Sigma_2, \delta_2, q_{0,2}, Q_{m,2})$*, l'automate* $G = (Q, \Sigma, \delta, q_0, Q_m)$ *est obtenu par composition parallèle de* G_1 *et* G_2 *tel que*
$G = G_1||G_2 = Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{0,1}, q_{0,2}), Q_{m,1} \times Q_{m,2})$ *où :*

- $Q = Q_1 \times Q_2$;
- $\Sigma = \Sigma_1 \cup \Sigma_2$;
- $\delta((x_1,x_2),\sigma) = \begin{cases} (\delta_1(x_1,\sigma), \delta_2(x_2,\sigma)) & \text{si } \delta_1(x_1,\sigma) \text{ et } \delta_2(x_2,\sigma) \text{ sont définies} \\ (\delta_1(x_1,\sigma), x_2) & \text{si } \delta_1(x_1,\sigma) \text{ est définie et } \sigma \notin \Sigma_2 \\ (x_1, \delta_2(x_2,\sigma)) & \text{si } \delta_2(x_2,\sigma) \text{ est définie et } \sigma \notin \Sigma_1 \\ \text{non défini} & \text{sinon} \end{cases}$
- $q_0 = (q_{0,1}, q_{0,2})$;
- $Q_m = Q_{m,1} \times Q_{m,2}$ ♦

La composition parallèle est également appelée « produit synchrone ». Ce nom vient du fait que les événements non communs à Σ_1 et Σ_2 peuvent se produire dans G dès qu'ils sont possibles dans G_1 ou G_2, alors que les événements communs ne peuvent se produire dans G que s'ils sont possibles en même temps dans G_1 et G_2. Cette opération est donc vue comme une synchronisation des événements communs.

La composition parallèle vérifie les propriétés de commutativité $G_1||G_2 = G_2||G_1$ et d'associativité $G_1||(G_2||G_3) = (G_1||G_2)||G_3 = G_1||G_2||G_3$, à condition de négliger le changement de nom des états qui passe de (x_1,x_2) à (x_2,x_1), ce qui est tout à fait acceptable puisque l'ordre des états d'origine dans le nom de l'état construit n'apporte aucune information. Par ailleurs, en considérant respectivement P_1 et P_2 comme les projections de Σ vers Σ_1 et Σ_2, les langages résultants de la composition parallèle sont :

- $L(G_1||G_2) = P_1^{-1}[L(G_1)] \cap P_2^{-1}[L(G_2)]$;
- $L_m(G_1||G_2) = P_1^{-1}[L_m(G_1)] \cap P_2^{-1}[L_m(G_2)]$.

La composition par produit (plus simplement appelée produit) permet quant à elle de n'obtenir que le comportement strictement synchrone, c'est-à-dire le comportement décrit par les événements communs aux deux automates. Ce comportement est donc un sous-ensemble du comportement décrit par la composition parallèle. Formellement, le produit est défini comme

suit :

Définition 12 (Produit)
Soient deux automates $G_1 = (Q_1, \Sigma_1, \delta_1, q_{0,1}, Q_{m,1})$ *et* $G_2 = (Q_2, \Sigma_2, \delta_2, q_{0,2}, Q_{m,2})$*, l'automate* $G = (Q, \Sigma, \delta, q_0, Q_m)$ *est obtenu par le produit de* G_1 *et* G_2 *tel que*
$G = G_1 \times G_2 = Ac(Q_1 \times Q_2, \Sigma_1 \cap \Sigma_2, \delta, (q_{0,1}, q_{0,2}), Q_{m,1} \times Q_{m,2})$ *où :*

- $Q = Q_1 \times Q_2$;
- $\Sigma = \Sigma_1 \cap \Sigma_2$;
- $\delta((x_1, x_2), \sigma) = \begin{cases} (\delta_1(x_1, \sigma), \delta_2(x_2, \sigma)) & \text{si } \delta_1(x_1, \sigma) \text{ et } \delta_2(x_2, \sigma) \text{ sont définies} \\ \text{non défini} & \text{sinon} \end{cases}$
- $q_0 = (q_{0,1}, q_{0,2})$;
- $Q_m = Q_{m,1} \times Q_{m,2}$ ♦

Le produit vérifie aussi les propriétés de commutativité $G_1 \times G_2 = G_2 \times G_1$ et d'associativité $G_1 \times (G_2 \times G_3) = (G_1 \times G_2) \times G_3 = G_1 \times G_2 \times G_3$, sous la même condition que la composition parallèle. Les langages résultants du produit sont :

- $L(G_1 \times G_2) = L(G_1) \cap L(G_2)$;
- $L_m(G_1 \times G_2) = L_m(G_1) \cap L_m(G_2)$.

Il est important de remarquer que si $\Sigma_1 = \Sigma_2$ alors $G_1 \times G_2 = G_1 || G_2$.

1.2.5 Automate non-bloquant

Un automate peut être bloquant. Cela signifie que cet automate peut générer des mots conduisant à un état non marqué dont il est impossible de sortir, il s'agit d'un *deadlock*, ou à un ensemble d'états qui peuvent générer des événements mais à partir desquels il sera toujours impossible d'atteindre un état marqué, il s'agit alors d'un *livelock*. Formellement :

Définition 13 (Automate bloquant)
Soit un automate G générant le langage $L(G)$ *et marquant le langage* $L_m(G)$*. Le langage de cet automate est bloquant si :*

$$L(G) \neq \overline{L_m(G)}$$

inversement, le langage est non bloquant si :

$$L(G) = \overline{L_m(G)}$$ ♦

Littéralement, un automate est bloquant s'il existe un mot de $L(G)$ qui n'est pas le préfixe d'un mot permettant d'atteindre un état marqué. Ceci correspond à un mot permettant d'atteindre un état non co-accessible. Par ailleurs, un état non accessible ne génère pas de mot du

langage $L(G)$. Par conséquent, l'utilisation conjointe des fonctions d'accessibilité (définition 8) et de co-accessibilité (définition 9) donnant un automate émondé conduit à un automate non-bloquant. En conclusion, un automate émondé est toujours non-bloquant.

1.3 Théorie de contrôle par supervision

1.3.1 Procédé et superviseur

La théorie de contrôle par supervision se base sur la théorie des langages (Section 1.2) et en particulier sur les langages rationnels (théorème 1) car il est souvent plus facile de représenter le comportement d'un système par un automate à état finis plutôt que par son langage. Le principe de cette théorie consiste à distinguer différents modèles, et notamment les modèles du procédé et le modèle du superviseur.

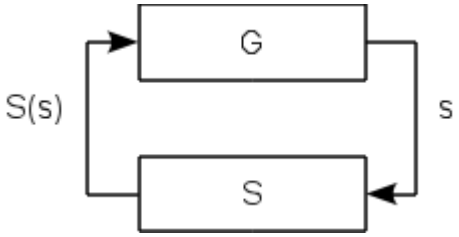

FIGURE 1.3 – Principe de contrôle par supervision

Le procédé est modélisé par un automate G générant le langage rationnel $L(G)$. Ce procédé représente l'ensemble des comportements possibles du système étudiés. L'objectif est de contrôler ce procédé et donc d'interdire certaines évolutions. Ceci revient à interdire des occurrences d'événements. Or, certains événements ne peuvent être interdits : une panne, une information provenant d'un capteur, une date d'échéance, etc. L'ensemble des événements Σ est donc partitionné en deux ensembles disjoints, l'ensemble des événements contrôlables Σ_c et l'ensemble des événements incontrôlables Σ_{uc}. Nous avons donc $\Sigma = \Sigma_c \cup \Sigma_{uc}$ et $\Sigma_c \cap \Sigma_{uc} = \emptyset$.

Le superviseur S est associé au procédé pour interdire les comportements non admissibles. Son rôle est donc d'interdire certaines occurrences d'événements, et bien sûr il ne peut interdire que des événements contrôlables. Ce superviseur est une fonction définie par $S : L(G) \rightarrow 2^{\Sigma}$. Ainsi, pour chaque mot s généré par G, le superviseur S lui renvoie l'ensemble des événements autorisés $S(s)$. G ne peut alors évoluer qu'en générant un événement inclus dans ce ensemble.

Le couple (S,G) représente donc le comportement du système sous contrôle. Ce système bouclé, noté S/G, génère le langage $L(S/G)$ et marque le langage $L_m(S/G)$. Ces deux langages sont déterminés grâce à la définition suivante :

Définition 14 (Langage généré et langage marqué par S/G)
Le langage $L(S/G)$ est défini récursivement comme suit :

1. $\varepsilon \in L(S/G)$
2. $[(s \in L(S/G))$ *et* $(s\sigma \in L(G))$ *et* $(\sigma \in S(s))] \Leftrightarrow [s\sigma \in L(S/G)]$

Le langage marqué $L_m(S/G)$ est défini par : $L_m(S/G) = L(S/G) \cap L_m(G)$ ♦

Ces langages ont la propriété d'inclusion suivante : $\emptyset \subseteq L_m(S/G) \subseteq \overline{L_m(S/G)} \subseteq L(S/G) \subseteq L(G)$. Ceci implique que si $L(G)$ est le comportement le plus permissif du système, le comportement sous contrôle permettant de réaliser les tâches souhaitées ($L_m(S/G)$) ne peut qu'être un sous-ensemble de ce comportement.

1.3.2 Contrôlabilité et superviseur

Étant donné un langage admissible L_a, est-il possible de déterminer un superviseur capable de restreindre le comportement de G à L_a ? Tout d'abord, il est bien sûr indispensable que ce Langage soit généré par G et qu'il permette de réaliser les tâches souhaitées. Ceci revient à dire que L_a doit être inclus dans le langage marqué de G : $L_a \subseteq L_m(G)$. La seconde condition est que ce langage soit contrôlable. Pour déterminer si un langage est contrôlable, il faut vérifier que la restriction de $L(G)$ à L_a n'amène pas à interdire des occurrences d'événements incontrôlables. Donc, pour tout mot $s \in L_a$ et pour tout événement $\sigma \in \Sigma_{uc}$, si $s\sigma$ est un mot de $L(G)$ alors ce doit aussi être un mot de L_a. Ceci est formalisé par la définition suivante :

Définition 15 (Contrôlabilité)
Soient L_a et $L(G)$ deux langages définis sur $\Sigma = \Sigma_{uc} \cup \Sigma_c$, avec Σ_c l'ensemble des événements contrôlables et Σ_{uc} l'ensemble des événements incontrôlables. L_a est dit contrôlable par rapport à G (ou à $L(G)$) et Σ_{uc} si et seulement si :

$$\overline{L_a}\Sigma_{uc} \cap L(G) \subseteq \overline{L_a}$$ ♦

Les conditions d'existence d'un superviseur marqué (i.e. permettant d'atteindre un état marqué de G) et non bloquant sont données par le théorème suivant :

Théorème 2 (Existence d'un superviseur marqué non bloquant)
Soit un langage admissible $L_a \subseteq L_m(G)$ tel que $L_a \neq \emptyset$. Il existe un superviseur marqué non bloquant S pour (L_a, G) tel que $L_m(S/G) = L_a$ si et seulement si L_a est contrôlable par rapport à G et à Σ_{uc}. ♦

Une démonstration de ce théorème peut être trouvée dans [Won09, p. 87]. Il est important de remarquer que, par définition, si ce superviseur existe, la propriété de non-blocage est vérifiée.

1.3.3 Suprême contrôlable

Que se passe-t-il si le langage admissible L_a est non contrôlable ? Il existe cependant des sous-langages de L_a contrôlables, et il est possible de définir l'ensemble les contenant tous :

Définition 16 (Ensemble des langages contrôlables de L_a)
L'ensemble de ces langages est noté $\mathcal{C}(L_a)$ et est défini par :

$$\mathcal{C}(L_a) = \{K \subseteq L_a \mid \overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}\} \quad \blacklozenge$$

Cet ensemble est non vide ($\{\} \in \mathcal{C}(L_d)$) et fermé par rapport à l'union. En particulier, cet ensemble contient un unique plus grand élément, appelé le suprême contrôlable de L_a, noté $L_a^{\uparrow c}$. Il est défini par :

Définition 17 (Suprême contrôlable de L_a)
Le suprême contrôlable de L_a, noté $L_a^{\uparrow c}$ est défini par :

$$L_a^{\uparrow c} = \bigcup\{L | L \in \mathcal{C}(L_a)\} \quad \blacklozenge$$

Littéralement, le langage suprême contrôlable de L_a s'obtient en faisant l'union de tous les langages de L_a contrôlables par rapport à G et à Σ_{uc}. Dans le pire des cas $L_a^{\uparrow c} = \emptyset$ puisque $\emptyset \in \mathcal{C}(L_a)$, et si L_a est contrôlable alors $L_a^{\uparrow c} = L_a$. [Won87] et [Cas07] présentent l'algorithme standard de calcul du suprême contrôlable :

Procédure 1

Étape 0 : *Soit $G = (Q, \Sigma, \delta, q_0, Q_m)$ un automate qui génère le langage L(G).*
Soit $E = (X, \Sigma, \xi, x_0, X_m)$ tel que $L_m(E) = K$ et $L(E) = \overline{K}$, et pour lequel $K \subseteq L(G)$.

Étape 1 *Soit $H_0 = (Y_0, \Sigma, \gamma_0, (q_0, x_0), Y_{0,m}) = G \times E$ avec $Y \subseteq Q \times X$.*
Tous les états de G doivent être considérés comme marqués pour déterminer $Y_{0,m}$.
Par hypothèse, $L_m(H_0) = K$ et $L(H_0) = \overline{K}$.
Les états de H_0 sont notés par des paires (q,x).
Posons $i = 0$.

Étape 2 : *Calculer :*

Étape 2.1 :

$$\begin{aligned} Y_i' &= \{(q,x) \in Y_i \mid \Gamma_G(q) \cap \Sigma_{uc} \subseteq \Gamma_{H_i}((q,x))\} \\ &\quad \text{avec } \Gamma_G(q) = \{\sigma \in \Sigma \mid \gamma_i(q,\sigma)!\} \\ &\quad \text{et } \Gamma_{H_i}((q,x)) = \{\sigma \in \Sigma \mid \gamma_i((q,x),\sigma)!\} \\ \gamma_i' &= \gamma_i | Y_i' \\ Y_{i,m}' &= Y_{i,m} \cap Y_i' \end{aligned}$$

Etape 2.2 :

$$H_{i+1} = Em(Y_i', \Sigma, \gamma_i', (q_0, x_0), Y_{i,m}')$$

Si H_{i+1} est un automate vide, i.e. (q_0, x_0) a été supprimé, alors $K^{\uparrow c} = \emptyset$ et STOP.
Sinon, poser $(Y_{i+1}, \Sigma, \gamma_{i+1}, (q_0, x_0), Y_{i+1,m}) = H_{i+1}$

Étape 3 : *Si $H_{i+1} = H_i$ alors $L_m(H_{i+1}) = K^{\uparrow c}$ et $L(H_{i+1}) = \overline{K^{\uparrow c}}$ et STOP.*
Sinon, poser $i \leftarrow i+1$ et aller à l'étape 2. ♦

Dans l'étape 2.1, Y_i' représente l'ensemble des états dans lesquels toutes les occurrences des événements incontrôlables se trouvent dans la fonction de transition de H_0.

L'automate H_{i+1} obtenu est donc non bloquant et il marque le suprême contrôlable de K, langage marqué de la spécification E.

1.3.4 Démarches de conception

Les sections précédentes nous ont permis de présenter quatre des modèles clés de la théorie du contrôle par supervision, à savoir le procédé G, le superviseur S, le langage admissible L_a et le suprême contrôlable de ce langage $L_a^{\uparrow c}$. Dans cette section nous allons montrer comment ces modèles sont utilisés pour résoudre le problème classique [1] qui pourrait s'exprimer sous la forme « Étant donné un procédé et des spécifications que doit respecter ce procédé, est-il possible de contrôler le procédé afin que son comportement soit restreint à ces spécifications ? ».

Répondre à cette question nécessite tout d'abord de modéliser le procédé. De manière naturelle, celui-ci est souvent décomposable en procédés élémentaires. Chacun de ces procédés a un comportement suffisamment simple pour être modélisable par un automate G_i ne comportant que quelques états, transitions et événements. Le procédé global est ensuite construit à partir de ces modèles élémentaires. L'opération utilisée est la composition parallèle qui permet d'obtenir un automate dont l'alphabet Σ est l'union des alphabets Σ_i des G_i, et dans lequel les comportements des différents composants sont synchronisés lorsqu'ils partagent des événements communs tout en permettant les occurrences des événements non partagés.

Il est ensuite nécessaire de modéliser les spécifications. Elles peuvent être également décomposés en spécifications élémentaires, chacune modélisée par un automate E_j. La plupart du temps, une spécification n'est écrite que sur un sous-alphabet de Σ_j de Σ. Si cela signifie que tous les événements de $\Sigma - \Sigma_j$ sont interdits, l'opération à utiliser pour construire la spécification globale E est le produit des spécifications E_j. Si ces événements sont autorisés, l'opération à utiliser est la composition parallèle. Une autre façon de faire est de mettre en boucle sur chaque état tous les événements de Σ autorisés afin que chaque spécification soit construite sur l'alphabet global. E est alors construit indifféremment par produit ou composition parallèle des

1. D'autres problèmes peuvent être abordés avec la théorie du contrôle par supervision : voir par exemple [Cas07] qui présente dans la section 3.4 deux autres problèmes.

E_i.

A la fin de ces deux opérations de modélisation, les automates G et E sont généralement construits sur le même alphabet Σ. L'automate $G \times E$ (ou indifféremment $G||E$) marque le langage admissible $L_a = L_m(G \times E)$. A partir de ce langage (*confer* les sections 1.3.2 et 1.3.3) il est possible de répondre à la question de la contrôlabilité et d'obtenir, si la réponse est négative, le plus grand sous-langage contrôlable de ce langage admissible. Si ce suprême contrôlable $L_a^{\uparrow c}$ existe, il est très important de comprendre qu'il permet de répondre par **l'affirmative** au problème posé et donc qu'il sera possible de déterminer un superviseur S tel que $L_m(S/G) = L_a^{\uparrow c}$.

Les questions concernant la réalisation de ce superviseur sous forme d'un automate, la minimalisation de ce superviseur [Su04] ou encore de génération de code [Vie06] sortent du cadre de cette thèse et ne seront donc pas détaillées ici.

1.4 Approches de décomposition

1.4.1 Les problèmes posés

La démarche de conception présentée dans la section précédente passe donc par la construction d'automates de petite taille, les G_i et les E_j, puis par la génération d'automates beaucoup plus importants, G, E et H tel que $L_m(H) = L_m(G \times E)^{\uparrow c}$. Cette génération pose deux problèmes, tous deux liés à la taille des modèles obtenus.

Le premier est l'explosion combinatoire. En effet, le produit ou la composition parallèle de deux automates respectivement de taille n et m donne dans le pire des cas un automate de taille $n \times m$. Le modèle K peut donc avoir $n \times m$ états si n représente le nombre d'états de G et m le nombre d'états de E, tandis que n et m vont croître exponentiellement avec le nombre de composants du procédé et de modèles de spécification. Ceci peut donc poser un problème de temps de calcul et de taille mémoire nécessaire, atténué par le fait que les logiciels actuels sont capables de calculer des automates comportant plusieurs centaines de milliers d'états.

Bien avant d'atteindre cette taille critique, le concepteur est confronté au problème d'interprétation des modèles [2]. En effet, interpréter un modèle ne comportant « qu'une petite centaine d'états » n'est pas toujours simple, la difficulté étant liée aussi bien au nombre d'états, qu'au nombre de transitions et à la structure de l'automate (boucles, nombres d'états adjacents pour chaque état...).

Les approches suivantes cherchent à résoudre ces deux problèmes.

2. Par interprétation, nous voulons dire ici la capacité du concepteur à comprendre quels sont les comportements modélisés par l'automate à états.

1.4.2 Approche modulaire

L'approche modulaire, illustrée figure 1.4, repose sur une décomposition des contraintes afin de créer plusieurs superviseurs au lieu d'un seul. Chaque superviseur représente ainsi une spécification unique et tous ces superviseurs locaux agissent simultanément pour restreindre le comportement du procédé [Nou04, Kom08].

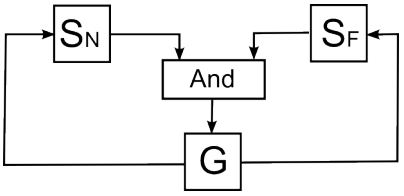

FIGURE 1.4 – Structure d'un système étudié par une approche modulaire

Sur la figure 1.4, le superviseur S est décomposé en deux superviseurs notés S_N et S_F. Ces deux superviseurs sont construits par rapport au procédé G et leur intersection forme le superviseur S_{NF} qui restreint le procédé. Chaque superviseur autorise ou interdit un ensemble d'événements et seul ceux qui ne sont interdits par aucun superviseur peuvent être générés.

Formellement, cela revient à dire que si S_{NF} est défini tel que $S_{NF} = S_N \cap S_F$, le contrôle modulaire doit être tel que :

- $L(S_{NF}/G) = L(S_N/G) \cap L(S_F/G)$
- $L_m(S_{NF}/G) = L_m(S_N/G) \cap L_m(S_F/G)$

Pour chaque spécification, il est possible d'obtenir un superviseur non bloquant maximum permissif. Cependant, pour obtenir un superviseur non bloquant à partir de la conjonction de deux superviseurs non bloquants, il faut en plus vérifier la propriété suivante :

Définition 18 (Superviseurs modulaires non bloquants)
Soient S_N et S_F deux superviseurs individuellement non bloquants pour G. $S_{NF} = S_N \cap S_F$ est non bloquant si et seulement si $L_m(S_N/G)$ et $L_m(S_F/G)$ sont non conflictuels, c'est-à-dire si et seulement si :

$$\overline{L_m(S_N/G) \cap L_m(S_F/G)} = \overline{L_m(S_N/G)} \cap \overline{L_m(S_F/G)} \qquad \blacklozenge$$

Les avantages de cette approche sont une interprétation plus facile grâce à la réduction de la taille des modèles S_i et S_i/G et un gain de temps et de ressources en calcul. Les deux inconvénient majeurs sont la nécessité de calculer le procédé G qui est de taille conséquente, et la vérification de non conflictualité des deux superviseurs.

1.4.3 Approche décentralisée

L'approche décentralisée, illustrée figure 1.5, reprend en partie l'approche modulaire dans la décomposition en petits superviseurs et propose en plus de réduire la taille du procédé sur lequel est construit chaque superviseur [Rud92]. En effet, sur de nombreux systèmes réels, le contrôle peut être scindé en plusieurs contrôleurs, chacun n'observant qu'une partie du système. L'idée est donc ici de construire, pour chaque superviseur local, une abstraction du système complet qui ne tienne compte que des événements de l'alphabet de ce superviseur. Cette abstraction se fait en utilisant la fonction de projection de l'alphabet global vers l'alphabet local.

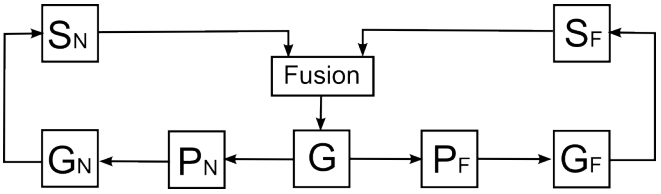

FIGURE 1.5 – Structure d'un système étudié par une approche décentralisée

Dans la plupart des travaux portant sur l'approche décentralisée [Tak98, Jia00], et comme dans l'approche modulaire, un événement du procédé ne peut avoir lieu que s'il est autorisé par tous les superviseurs. Les auteurs de [Yoo02] ont généralisé cette approche en proposant une structure de contrôle dans laquelle les superviseurs peuvent être fusionnés par *conjonction* ou par *disjonction*. Dans une fusion conjonctive, un événement n'est autorisé sur le procédé G que s'il n'est interdit par aucun des superviseurs. Dans une fusion disjonctive, un événement est interdit sur le procédé G que s'il n'est autorisé par aucun des superviseurs.

Cependant, comme pour l'approche modulaire et quelle que soit la structure de fusion choisie, une décomposition par approche décentralisée est soumise au problème de blocage dans le cas où la condition de non-conflictualité des superviseurs n'est pas vérifiée.

Cette approche permet donc elle aussi de réduire la taille des modèles, y compris les modèles du procédé. Elle facilite donc la compréhension et réduit le risque d'explosion combinatoire pour les modèles de spécification et de procédé sous contrôle. Cependant, le modèle G reste à calculer globalement avant de déterminer ses projections sur les alphabets locaux.

1.4.4 Approche hiérarchique

Le principe de l'approche hiérarchique repose sur une conception des modèles à plusieurs niveaux. Les auteurs de [Zho90, Cha03] proposent un système décomposé en deux niveaux. La figure 1.6 illustre ce découpage dans lequel les modèles d'indice l sont ceux de bas niveau et ceux d'indice h ceux de haut niveau.

Le procédé de bas niveau G_l est le modèle de l'ensemble du procédé, alors que G_h n'est qu'une abstraction de ce modèle. G_h évolue sur un alphabet de haut niveau généré par G_l. Cet

alphabet est l'alphabet de sortie de G_l qui est en fait une machine de Moore : une fonction de transition de sortie associe à chaque état de G_l un événement de l'alphabet de sortie, dont un événement particulier est l'événement de silence (associé à tous les états de G_l qui ne doivent pas faire évoluer G_h).

Le modèle S_l de son coté représente le superviseur qui contrôle G_l, alors que S_h contrôle G_h. Un canal d'information, inverse du canal d'information représenté par la fonction de transition de sortie de G_l, existe aussi du superviseur de haut niveau S_h vers le superviseur de bas niveau S_l. Ce canal permet de restreindre le comportement de niveau bas à partir des restrictions de niveau haut.

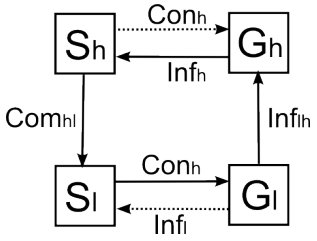

FIGURE 1.6 – Structure d'un système étudié par une approche hiérarchique

L'approche hiérarchique nécessite cependant l'introduction de la notion de *consistance hiérarchique* qui permet de vérifier la cohérence entre les restrictions de niveaux bas et haut, ceci afin que le comportement obtenu soit aussi permissif que celui qui aurait été obtenu par l'approche classique.

Cette approche permet elle aussi de réduire la taille des modèles, sauf celle du procédé G_l qui a la même taille que G dans l'approche classique. Elle peut par ailleurs être utilisée conjointement avec l'approche modulaire ou décentralisée [Cha00, Fen08].

1.5 Conclusion

Les systèmes à événements discrets se prêtent bien à la problématique des modes. Dans ce chapitre, nous avons présenté les bases de la théorie du contrôle par supervision et des outils mathématiques utilisés, la théorie des langages et les automates à états. Nous avons insisté sur les concepts de procédé, spécification, procédé sous contrôle, contrôlabilité et non blocage qui seront utiles pour la compréhension de nos travaux. Nous avons aussi exposé les principes de quelques approches cherchant à résoudre les problèmes d'interprétation et de calculabilité liés à la taille des modèles. Cependant, ces approches ne sont pas applicables pour la gestion de modes, car en construisant globalement le procédé G elles considèrent l'utilisation systématique de tous les composants du système. Nous ferons référence à ces travaux pour justifier notre approche dans le chapitre 3.

2

Approche modale

Sommaire

2.1 Introduction

Nous retiendrons ici que l'approche modale est le fait de décrire le fonctionnement d'un système en utilisant le concept de mode. Un mode sera vu comme un fonctionnement non permanent du système. Ce fonctionnement a un début, une fin, et réalise une fonction au sens large. La succession de modes correspond alors à une succession de fonctionnements non permanents du système tel que, pris ensemble, ils correspondent à un fonctionnement continu et permanent de celui-ci.

La première difficulté dans l'utilisation de l'approche modale se trouve dans l'identification de ces modes et d'en déduire des modèles lors de la conception d'un système. Il est alors nécessaire de procéder à une discrimination des données du cahier des charges pour se focaliser uniquement sur celles utiles à chaque fonctionnement non permanent du système. L'ensemble de ces informations, pour un mode donné, caractérise une configuration particulière du système.

La deuxième difficulté dans cette approche est la succession des modes et plus précisément les transitions entre modes. En effet, si les fonctionnements non permanents du système sont connus, par le cahier des charges, qu'en est-il des fonctionnements du système entre ses modes ? De plus, quelles sont les informations nécessaires pour caractériser complètement une reconfiguration, c'est-à-dire un changement de configuration ?

Ce chapitre présente différents travaux qui portent sur la conception de modes et/ou sur la gestion des modes. Bien que tous les travaux ne traitent pas de tous les problèmes en profondeur, nous avons sciemment relevé les points qui nous semblent importants. Ceux-ci concernent la définition d'un mode telle qu'explicitée par les auteurs, les propositions soumises, les langages, méthodes et théories utilisés ou encore l'exhaustivité des modes et des commutations prise en compte.

Une synthèse de tous ces points, pour chacun des travaux, est présentée en fin de chapitre. Cette synthèse nous permet de regrouper toutes les informations afin d'être en mesure de définir les notions abordées et d'exprimer nos objectifs quant à la proposition de ces travaux.

2.2 Le GEMMA

Le GEMMA, pour Guide d'Étude de Modes de Marches et d'Arrêts, a été proposé par l'ADEPA [ADE81]. Il est destiné à être utilisé dans le monde industriel lors de la conception d'un système automatisé de production (S.A.P). Il repose sur le concept d'un système automatisé décomposable en trois parties :

- une partie opérative (P.O.) qui regroupe les mécanismes, actionneurs et capteurs du système physique ;
- une partie relation (P.R.) qui regroupe les capteurs-opérateurs, composants de signalisation, visualisation et de communications. Elle correspond au pupitre de commande de

l'opérateur ;

– une partie commande (P.C.) qui regroupe tous les composants permettant le traitement de l'information en provenance des parties opérative et relative.

Le GEMMA fut proposé pour aider à représenter les différents modes d'un système pour faciliter la conception de celui-ci. En effet, la décomposition abstraite d'un système en modes est couramment utilisée pour réduire la complexité d'un système. Cependant, aucune méthodologie n'existait pour utiliser cette décomposition durant les phases de conception. De plus, lorsqu'un système a plusieurs modes, des problèmes peuvent apparaître lors de commutations entre eux. Cela vient de la démarche de décomposition. Dans cette démarche, le concepteur représente les commutations du système d'après l'image qu'il se fait de celui-ci lors de l'occurrence d'un événement de commutation. Ceci correspond à une description du contrôleur car l'état réel du système n'est pas connu mais il est supposé. Les problèmes apparaissent lors d'une incohérence entre l'état supposé du système par le concepteur et l'état réel du système physique.

Le GEMMA proposé est un outil graphique structuré, illustré sur la figure 2.1, d' *aide à la conception*. Ce guide aide le concepteur à définir, à partir d'un canevas type, les différents modes d'une machine ainsi que les passages entre ces modes. Le canevas est composé de deux zones principales. La zone de gauche, appelée *PZ*, correspond à l'état inopérant de la partie commande vis-à-vis de la partie opérative. La seconde zone, celle de droite, correspond au fonctionnement de la partie commande et regroupe trois familles de procédures. La famille *F* regroupe les procédures de fonctionnement (de la partie opérative). Elle assure la mise en service, ou hors-service, de la production normale et effectue les vérifications de production. La famille *A* est constituée des procédures d'arrêt et de remise en route de la partie opérative. Elle assure le fonctionnement du système lorsqu'un arrêt est demandé ou exigé. Enfin, la famille *D* concerne les procédures de défaillances. Ces procédures ont pour objectifs de limiter les conséquences d'une défaillance sur le personnel et le matériel.

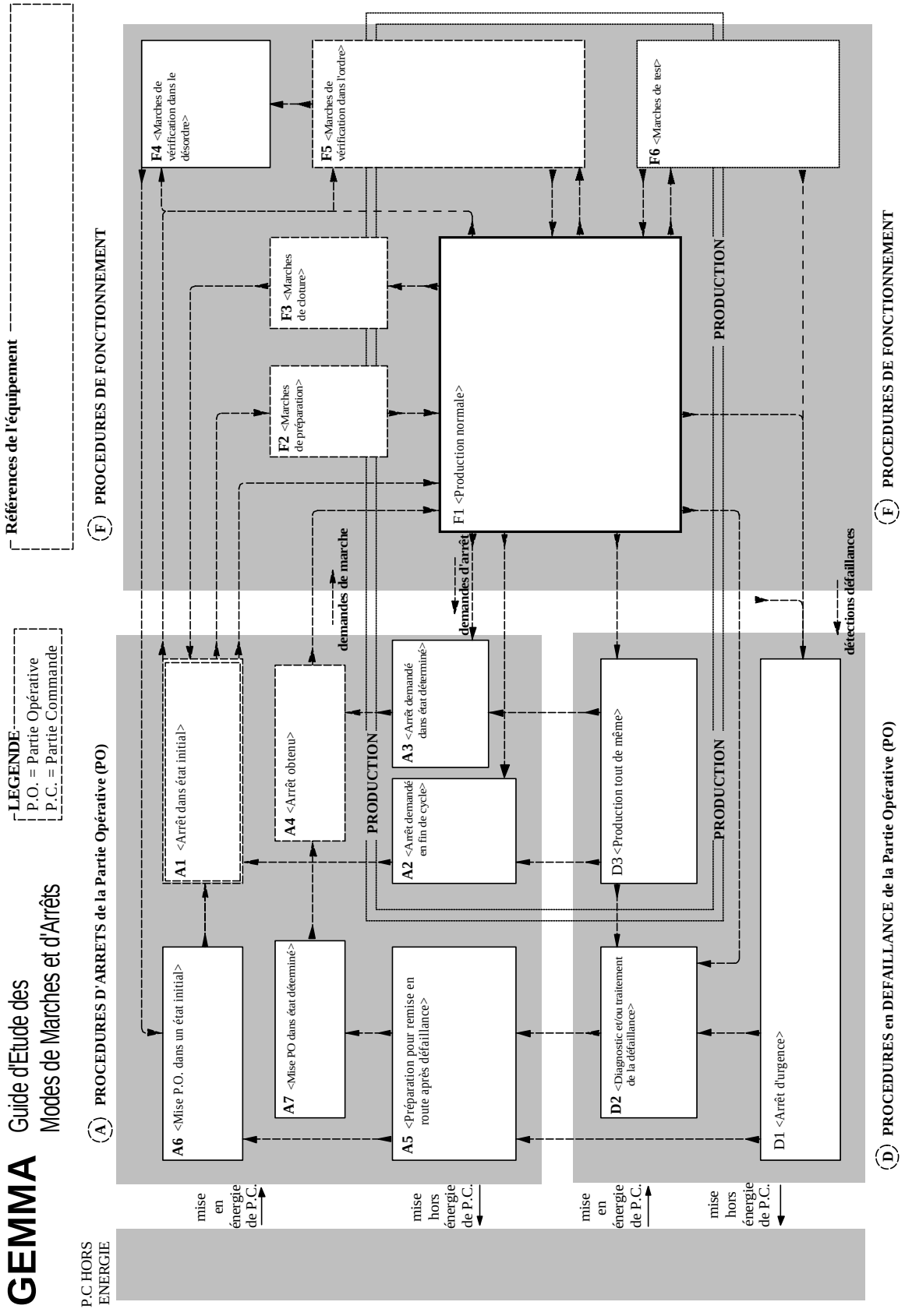

FIGURE 2.1 – LE GEMMA

Démarche d'utilisation

Une démarche d'utilisation pour le GEMMA est proposée dans [Mor97]. Elle se décompose en quatre phases :

- La première phase consiste à sélectionner les modes utilisés pour la conception et supprimer les autres [1] ;
- La seconde est la détermination des conditions de commutation entre les modes sélectionnés ;
- La troisième phase est la conversion du modèle GEMMA en un modèle final implémentable. Ce modèle final est formalisé en GRACFET et est considéré comme un GRACFET *maître* contrôlant l'activation et la désactivation des modes du système ;
- La dernière est celle de la création d'un modèle GRACFET pour chacun des modes sélectionnés et représentant l'action à accomplir. Chacun de ces modèles doit être synchronisé avec le GRACFET maître.

D'un point de vue opérationnel, le GEMMA est attractif car il fournit une aide pour le concepteur qui s'appuie sur une présentation graphique. Cette représentation l'aide à éviter les fautes de conception en le guidant dans ses choix lors de la sélection de modes et des commutations entre ceux-ci.

Cependant, ce guide est limité vis-à-vis de la complexité des systèmes industriels actuels. Ces systèmes comportent souvent plusieurs machines qui sont difficiles à toutes représenter sur un même graphique. Ceci est dû au fait que le GEMMA considère les parties commande et opérative comme une entité unique et indivisible. Or une partie commande peut contrôler plusieurs parties opératives totalement indépendantes et ayant potentiellement des modes différents. Il est alors impossible de faire une distinction entre des commandes locales propres aux machines et une commande globale conçue par le GEMMA.

Le GEMMA n'est également basé sur aucune méthode mathématiquement définie. Il est une démarche développée empiriquement par les concepteurs. Ce guide ne contient donc aucun mécanisme de vérification permettant d'assurer que les ordres de commutations soient tout le temps corrects [Sto05].

2.3 Modecharts

Les auteurs du formalisme des Modecharts s'inspirèrent de celui des Statecharts [Har87] en partant du principe que *"le concept de modes est familier pour concevoir un système de contrôle de processus"* [Jah88, Jah94, Puc95]. Dans ces travaux, un mode est vu comme une boite noire réalisant une fonction. Reposant sur une sémantique de logique temps-réel (RTL), ce formalisme est donc approprié pour représenter des systèmes découpés fonctionnellement

1. Tous les modes exprimés dans le GEMMA ne sont pas nécessaires pour l'étude de tous les systèmes.

en modes tels que les systèmes automatisés, les systèmes embarqués ou les systèmes hybrides. Les modes peuvent être vus, pour les auteurs, comme différentes partitions de l'espace d'états d'un système et le concept de modes est un moyen efficace pour représenter des spécifications modulaires d'une machine de taille importante. La figure 2.2 illustre un système composé par plusieurs modes.

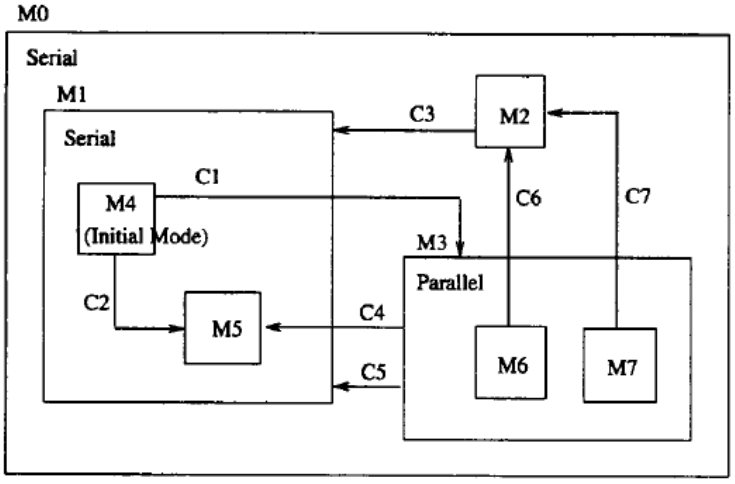

FIGURE 2.2 – Conception d'un système composé par des modes

Pour les auteurs, l'étude d'un système par ses modes de fonctionnement impose une méthodologie de conception lorsque plusieurs modes sont connectés entre eux afin de ne pas compromettre le fonctionnement de ce système. Les Modecharts restent une représentation abstraite de décomposition d'un système en modes. Ce formalisme utilise comme bloc élémentaire un mode, et ne crée un mode que par la composition d'autres modes. Bien que les auteurs ne définissent pas comment construire ces blocs élémentaires à partir des composants d'un système, cette composition se rapproche d'une description du comportement de la partie commande du système où l'interconnexion des modes représente le comportement global. Ce comportement s'obtient alors par la prise en compte et l'interconnexion de tous les modes voulus.

Les auteurs traitent dans ces travaux de la difficulté de représenter un système par ses modes de fonctionnement et notamment, des erreurs provenant de l'interconnexion de modes entre eux. Nécessitant des vérifications pour éliminer ces erreurs, les auteurs proposent un formalisme pour construire un modèle représentant les modes d'un système et les liens qu'il y a entre eux. Les avantages de ce formalisme sont d'assurer l'absence de fautes de conceptions telles que des fautes d'incohérences, d'indéterminisme, etc. et de permettre une représentation hiérarchique du système.

La proposition des auteurs s'articule autour de deux phases de construction. La première concerne les connexions entre modes, la deuxième la composition de modes : du point de vue des connexions entre modes, celles-ci doivent être instantanées pour éviter l'indéterminisme

provoqué par l'arrivée d'un événement lors d'un changement de mode. Les auteurs définissent deux types possibles de connexions entre modes : série et parallèle. Les modes séries correspondent à un fonctionnement séquentiel du comportement où le système évolue dans un mode un certain temps avant que ne soient validées les conditions autorisant le passage au mode suivant. Les modes parallèles correspondent à un comportement simultané de plusieurs modes. Par exemple, sur la figure 2.2, le mode M_1 est un mode série car les sous-modes M_4 et M_5 fonctionnent l'un après l'autre. Le mode M_3 est un mode parallèle car les sous-modes M_6 et M_7 fonctionnent simultanément. Dans ce mode, pour éviter de l'indéterminisme, aucune transition ne doit exister entre ces deux modes. Si une telle transition existe, le système pourrait évoluer vers un mode dans lequel il est déjà [2].

Concernant la composition de modes, les auteurs établissent certaines propriétés à vérifier dans le cas où un mode est construit par composition de plusieurs autres. Un mode qui n'est pas construit par la composition d'autres modes est un mode élémentaire, appelé *mode primitif*. Comme dit précédemment, le comportement de ce mode n'est pas décrit et doit être défini par le concepteur. De ce mode primitif, les auteurs définissent les notions de mode composé, de mode racine, et de mode fils.

Définition 19 (Caractérisation d'un mode)

- *un mode M_i est un mode primitif ssi* $\forall j, M_j \not\subset M_i$
- *M_i est un mode composé ssi* $\exists j, M_j \subset M_i$
- *M_i est un mode racine ssi* $\forall j, M_i \not\subset M_j$
- *M_i est un fils direct du mode M_j (et inversement) ssi :*

$$M_i \subset M_j \wedge (\forall k, j \neq k \wedge M_i \subset M_k \rightarrow M_j \subset M_k)$$ ♦

Pour cette définition, en considérant la figure 2.2, les modes primitifs sont les modes M_2,M_4,M_5,M_6 et M_7 car ils ne contiennent aucun autre mode. M_0 est un mode racine car il n'est contenu dans un aucun autre mode. Les mode M_0,M_1 et M_3 sont des modes composés avec, par exemple, M_1 et M_3 les modes fils du mode M_0.

À partir de ces notions, les auteurs introduisent la propriété *bien-formé*. Cette propriété est vérifiée récursivement sur les modes de la manière suivante :

Définition 20 (Mode* bien-formé*)

1. *Un mode primitif est bien-formé.*
2. *Supposons que $M_1, M_2, ..., M_n$ soient des modes racines bien-formés, et qu'au plus un mode M_i soit noté comme mode initial. Alors un mode série M ayant tous ces modes comme fils direct est un mode* bien-formé.

2. Par exemple, si une transition allant de M_6 à M_7 existe, les modes M_2 et M_7 s'activeraient lors de la désactivation de M_6. Or, le mode suivant M_7 est justement le mode M_2. Le système ne peut commuter vers M_2 car le fonctionnement de ce mode est déjà en cours.

3. *Supposons que $M_1, M_2, ..., M_n$ soient des modes racines bien-formés, qu'aucun ne soit initial et qu'ils ne soient pas connectés entre eux. Alors un mode parallèle M ayant tous ces modes comme fils direct est un mode* bien-formé.
4. *Supposons un mode M composé par deux modes bien-formé M_i et M_j. Le mode M est bien-formé si M_i et M_j sont connectés en série et qu'un seul des deux est initial.* ⧫

Littéralement, un mode *bien-formé* est soit un mode n'en contenant aucun autre, soit un mode ne contenant que des modes bien-formés. De plus, un mode série bien-formé doit n'avoir qu'un seul mode initial alors qu'un mode parallèle bien-formé ne doit en avoir aucun.

Une conséquence de cette définition est que pour valider un mode comme bien-formé, la vérification s'effectue en premier par les modes primitifs et remonte jusqu'aux modes racines. Cela correspond à une vérification ascendante.

Cependant, ces définitions ne font pas référence aux informations nécessaires pour déterminer les conditions initiales de chaque mode, c'est-à-dire le ou les modes initiaux. La détermination des modes initiaux est ainsi définie :

Définition 21 (Mode initial)
Un mode M est un mode initial *si* au moins *une des conditions suivantes est remplie :*

- *M est un mode racine,*
- *si son parent M' est un mode série et que M soit son mode initial,*
- *si son parent M' est un mode parallèle.* ⧫

En conclusion, à travers ces trois définitions, le formalisme Modecharts permet d'assurer qu'il n'y a pas d'erreurs de conception dans l'interconnexion des modes entre eux à condition que ces modes soient bien-formés. Cependant ces travaux partent de l'hypothèse que le mode primitif, bloc élémentaire de composition, est au préalable bien-formé. Si ce n'est pas le cas, alors le respect des règles de construction n'assure pas l'inexistence d'erreurs. La partie délicate est donc la conception des modes primitifs, d'autant plus qu'aucune piste de réflexion n'est donnée pour les concevoir.

2.4 Mode-automata

Un Mode-automata est un formalisme proposé dans [Mar92, Mar98] qui utilise la théorie des automates à états pour représenter le comportement modal d'un système et un langage impératif de programmation pour les tâches que le système doit réaliser. Les auteurs définissent un mode comme « *un ensemble d'états dans lequel le système peut évoluer un certain temps avant de basculer dans un état n'appartenant pas à cet ensemble* ». Ce formalisme est destiné aux concepteurs de systèmes réactifs synchrones.

Les auteurs font le constat qu'il n'existe pas de formalisme qui permet à partir d'un modèle de valider des propriétés et qui soit facilement et directement implémentable. En effet, la plupart des formalismes d'étude pour la conception sont utilisés seulement pour améliorer la compréhension du système et ne sont pas directement ceux implémentés. Il y a alors une phase de conversion de modèles qui est nécessaire. Inversement, les formalismes d'implémentation ne permettent ni une représentation de systèmes complexes, ni de faire de la validation. Ainsi, une propriété validée dans une loi de commande représentée dans un formalisme non implémentable peut ne plus être validée dans un formalisme implémentable suite à l'étape de conversion. Ainsi, l'objectif des auteurs dans ces travaux est d'obtenir une démarche pratique (et non uniquement théorique) permettant de concevoir un modèle directement implémentable d'une loi de commande d'un système complexe. Le formalisme proposé doit permettre de représenter la notion de hiérarchie, un comportement modal, et être adéquat pour effectuer des phases de simulation et de vérification du comportement en vue d'une validation des spécifications.

Les auteurs ont précédemment travaillé sur les langages synchrones tels que *Lustre* et *Signal* qui sont des langages implémentables [Cas87, Hal91, LeG91]. Ces langages permettent également de faire de la validation de spécifications. Les auteurs proposent alors, en se basant sur *Lustre*, d'ajouter un module utilisant le formalisme des automates à états pour y incorporer un comportement modal et une hiérarchie [Mar98, Mar03].

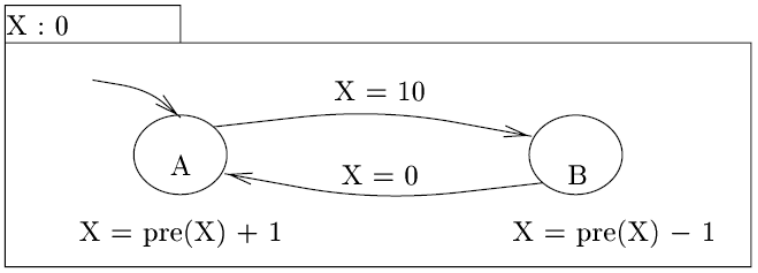

FIGURE 2.3 – Exemple d'un système représenté par un 'mode-automata'

La figure 2.3 illustre l'extension représentant un *Mode-automata* pour un système ayant deux modes de fonctionnement A et B. Cet automate décrit un programme qui émet un entier X de valeur initiale nulle. Le mode A est un mode d'incrémentation alors que le mode B est un mode de décrémentation. La fonction $pre()$ est propre au langage *Lustre* et correspond aux valeurs précédentes de X sur le flots de données $X(n)$. Le changement de mode se fait suivant la valeur de X. Dans chaque mode, c'est le langage impératif Lustre qui est utilisé pour manipuler les variables ou exécuter des actions. Chaque mode correspond ainsi à une fonction où un jeu d'instructions est à réaliser jusqu'à ce que les conditions de sorties soit validées.

Formellement, un Mode-automata est un septuplet (Q, q_0, I, O, L, f, T) où :

– Q est l'ensemble de modes et q_0 le mode initial ;

- I et O sont respectivement les ensembles des variables d'entrée et de sortie, et sont disjoints ;
- $L : O \rightarrow \mathbb{Z}$ est une fonction définissant la valeur initiale des sorties ;
- T contenu dans $Q \times (I, L) \times Q$ est l'ensemble des transitions labélisées par des conditions de transition ;
- f : est une fonction permettant de décrire les instructions (à réaliser) dans chaque états.

Les auteurs font remarquer que le modèle Mode-automata est équivalent à une machine de Moore à laquelle on a ajouté la fonction $pre(X)$ et les équations à réaliser dans chaque états.

En conclusion, les avantages de la démarche proposée sont de permettre la conception d'un modèle représentant le comportement modal d'un système incluant des instructions en langage impératif qui sont facilement implémentables. Il est également possible, du fait du langage choisi, de procéder à des phases de simulation et de validation. Néanmoins, il est nécessaire de respecter certaines hypothèses. La première est celle de l'unicité de mode actif. Le système n'est considéré que dans un seul mode actif à la fois. Dans le cas inverse, différents jeux d'instructions seraient exécutés. Ceci pourrait causer des problèmes d'affectations de variables. De plus, à travers les exemples étudiés, la démarche proposée semble être destinée aux systèmes ayant une très forte séquentialité d'exécution de modes bien que cette limite ne soit pas explicitée par les auteurs. De manière plus générale, les auteurs expriment une vision particulière d'un mode comme étant la réduction [3] d'un programme complet. Cette réduction représente le comportement du système global restreint au mode considéré. Bien que totalement en accord sur le principe, nous remarquons que les auteurs n'ont pas exprimé de démarche ou de définition particulière permettant de déterminer ce qui correspond à la projection d'un programme, hormis celui du fonctionnement local d'un programme plus global. Pour finir, les auteurs remarquent un problème de *perte d'information* lors d'un changement de mode. Cette perte d'information correspond à une information présente dans le mode source mais absente du mode cible et est due à l'activation d'un mode dans son état initial « *quels que soient les événements qui se sont précédemment passés* ». Le mécanisme de commutation entre modes est par conséquent source d'erreurs de conception s'il y a une perte d'information. Les auteurs insistent alors sur le fait que, pour changer de modes, l'accent doit être mis sur l'information qui est transmise. Ces erreurs sont normalement identifiables lors des phases de vérification. Cependant, à cause de la taille des modèles de la théorie des automates, ces phases sont longues et peuvent ne pas être exhaustives. Une solution envisagée est l'utilisation d'outils donnant des modèles finaux sûrs par construction en vue de réduire les phases de vérifications.

3. les auteurs utilisent le mot "projeté" comme fonction de réduction

2.5 Spécifications Robustes de conception

Le laboratoire IRCCyN, rattaché à L'école Centrale de Nantes, a proposé une démarche de conception de spécifications afin d'éviter les fautes (explicites ou implicites) du cahier des charges [Rak05]. Cette démarche est utilisée pour l'étude de systèmes électroniques représentés par des automates et qui manipulent des données binaires.

Les auteurs cherchent à résoudre des problèmes de sûreté de fonctionnement et plus particulièrement liés aux fautes qui proviennent d'un manque de clarté ou d'un défaut implicite dans les spécifications. Dans le cas de systèmes à plusieurs modes de fonctionnement, des fautes apparaissent également lors d'un changement de modes ; le système sort d'un mode mais n'entre dans aucun autre mode ; le système se bloque car il est dans un état indéterminé.

L'objectif poursuivi par les auteurs est la recherche de la complétude des spécifications assurant que le système modélisé est déterministe dans son comportement et lors d'un changement de modes, et ce, indépendamment de la valeur des données. La proposition se compose d'une phase de modélisation et d'une phase de vérification des propriétés de complétude et de cohérence des spécifications du cahier des charges. L'exemple directeur traité est celui d'un système de pilotage d'une fonction électronique automobile.

En s'inspirant de travaux sur la gestion de modes tels que [Jah94, Mar98], les auteurs définissent les modes d'un système comme « *les différentes variantes de fonctionnements qui peuvent être activées au cours de sa vie. Deux modes diffèrent entre eux par leur contenu fonctionnel, la qualité de service, la nature des données consommées ou fournies, ou encore du type de connexions qui les relient entre eux* ».

Un mode regroupe les états du système pendant son fonctionnement dans une phase opératoire identifiée – telle que nominale ou dégradée. Les auteurs considèrent que « *si chaque phase opératoire est assimilée à un mode, alors le comportement complet du système peut être appréhendé comme l'union des comportements de tous ses modes* ». La complétude est donc la propriété validant que tous les comportements possibles du système sont bien exprimés. De plus, si chaque mode représente un comportement du système, il convient alors de considérer que tous « *Les modes d'un système doivent s'exclure mutuellement dans le temps* » ; c'est la propriété d'unicité de mode actif qui implique de l'indéterminisme si elle n'est pas respectée.

Les étapes de modélisation et de vérification sont réalisées chacune par une fonction. La première fonction, appelée *combinaison modale*, identifie toutes les entrées et les sorties de chaque mode. Chacun possède théoriquement 2^{N+P} états avec N et P respectivement le nombre d'entrées et de sorties. Cette identification se fait par la recherche d'une correspondance entre entrée-sortie où chaque état est représenté par une seule entrée et une seule sortie. Tous les états sont pris en compte en recherchant toutes les combinaisons binaires des entrées/sorties possibles. De l'ensemble de ces états identifiés, le concepteur doit sélectionner ceux qui constituent l'ensemble des états initiaux de chaque mode – ceux dans lesquels le système peut être

en entrant dans ce mode. L'ensemble des modes doit couvrir toutes les évolutions *possibles et impossibles* du système global. Les évolutions impossibles sont celles des occurrences, dans un état donné, de combinaisons d'entrées qui sont *technologiquement* impossibles. Une situation peut aussi être qualifiée d'impossible par le concepteur s'il estime que la *probabilité* d'occurrence d'une combinaison d'entrées, liée à la gravité de ses conséquences, mérite qu'elle soit rejetée de l'univers du possible. Tous les états correspondant à des évolutions impossibles sont alors regroupés dans des modes défauts, ou modes replis, représentant un fonctionnement *fictif* du système.

La seconde fonction, appelée *saturation*, se concentre sur les connexions entre modes. Le but de cette fonction est d'assurer qu'aucune transition entre modes n'est oubliée, ce qui mènerait le système dans un état indéterminé. La fonction de saturation peut s'utiliser de deux façons différentes. La première revient à créer une transition de chaque mode vers tous les autres modes. Ainsi, toutes les possibilités (de commutation) sont prises en compte. Ensuite, et suivant les cas, les transitions n'arrivant (technologiquement) jamais peuvent être supprimées. La deuxième manière ne crée pas toutes les commutations possibles mais laisse au concepteur le choix des connexions qu'il souhaite représenter. La saturation s'assure alors que toutes les entrées possibles (technologiquement) aient bien une réaction dans le système.

En conclusion, en ce qui concerne la gestion de modes, ces travaux apportent une démarche claire qui effectue une recherche exhaustive des comportements pouvant se produire. Les auteurs partent ainsi d'une description opérative où tous les états possibles du système sont représentés. La proposition apportée par l'auteur permet de combler un manque dans la sûreté de fonctionnement en corrigeant des fautes implicites et non détectées dans le cahier des charges. Elle permet également d'éviter de pallier des *défaillances* auxquelles le concepteur n'aurait pas pensé. Néanmoins, le choix de réaliser une construction exhaustive sur l'ensemble des combinaisons binaires implique un problème d'explosion combinatoire. Cependant, les auteurs relèvent le fait que c'est la seule solution pour n'oublier aucun comportement possible. La difficulté réside alors dans la possibilité de réaliser une construction exhaustive qui permet également de limiter l'explosion combinatoire.

2.6 CASPAIM : démarche de gestion de modes

Une démarche de conception, nommée CASPAIM, pour les systèmes automatisés de production ayant des modes de fonctionnement est proposée au LAGIS [Dan00, Dan02]. Les objectifs de l'équipe de recherche sont l'augmentation de la sûreté de fonctionnement et l'utilisation d'un formalisme permettant d'implémenter des solutions expérimentales.

L'équipe s'est inspirée de formalismes existants [Har87, Jah94, Mar92] et du GEMMA proposé par l'ADEPA [ADE81]. Cependant, le GEMMA ne peut être utilisé tel quel à cause de la complexité des systèmes actuels (section.2.2). Les auteurs désirent également mettre en œuvre

un formalisme existant et utilisable pour implémenter les modèles conçus. Leur choix s'est porté sur le formalisme des SyncCharts [And96] basé sur le langage Esterel [Ber92]. À l'origine de ce choix, les auteurs ont besoin d'un outil permettant de représenter une décomposition, une hiérarchie et des *préemptions* de transitions. Ils ont également la nécessité de valider le comportement spécifié de leurs modèles.

Les auteurs proposent une démarche pour *déterminer les modes* d'un S.A.P. Son but est d'être utilisable pour une grande majorité de systèmes complexes. La complexité d'un système est définie par l'hétérogénéité de ses composants, de la flexibilité qu'il offre et par les nombreux paramètres qui peuvent être manipulés.

Pour atteindre cet objectif, les auteurs effectuent, sur la base du GEMMA, une caractérisation des états possibles des ressources du système. Les auteurs distinguent les ressources de production dans trois états possibles. Les ressources *engagées en production* sont celles utilisées pour réaliser la tâche en cours ; les ressources *en attente* sont celles qui peuvent être utilisées en cas de défaillance d'une des ressources engagées ; les ressources *à l'arrêt* sont celles qui sont inutilisables pour cause de défaillance ou hors tension.

Les auteurs proposent également une caractérisation du changement de modes, appelé *reconfiguration*. Une reconfiguration désigne une des phases du traitement d'erreur qui permet de substituer un état exempt d'erreur à l'état erroné. Trois types sont identifiés : la reconfiguration mineure ne concerne que les ressources engagées et consiste à répartir sur ces ressources les tâches/opérations qui auraient dû être exécutées sur les ressources défaillantes. Elle se traduit par la désactivation des ressources en panne et un nouveau paramétrage de la commande. Le deuxième type est la reconfiguration significative qui concerne les ressources engagées et en attente. Elle est mise en œuvre en cas d'échec d'une reconfiguration mineure. Elle consiste au remplacement des ressources défaillantes par des ressources en attente. Enfin, la reconfiguration majeure concerne toutes les ressources du système et est mise en œuvre pour atteindre au plus près les objectifs, quitte à utiliser des ressources partiellement défaillantes.

À partir d'une caractérisation des ressources selon les objectifs à atteindre, les auteurs déterminent des points de vue du système. Chaque point de vue est un état de fonctionnement du système et les reconfigurations sont alors des changements de point de vue.

Cette approche par point de vue permet de fournir une démarche de décomposition en considérant qu'un ensemble de composants et d'objectifs sont décomposables en sous-ensembles. Ainsi, pour les auteurs, « *décomposer en point de vue revient à se demander quelles sont les informations dont nous avons besoin pour caractériser le comportement du système à un niveau de décomposition donné* ». La décomposition successive en points de vue est poursuivie jusqu'à l'obtention d'un point de vue non décomposable, c'est-à-dire qui ne correspond qu'à une seule interprétation possible. Il y a ainsi une encapsulation des points de vue.

Cette décomposition est par conséquent une décomposition *descendante* du système, où les

auteurs partent d'une vue contrôle du système, jusqu'à la sélection des composants (physiques) élémentaires du système. Après décomposition, les auteurs se retrouvent avec une représentation d'un système en plusieurs modèles hiérarchiques. Ces modèles possèdent à la fois une description du comportement de la partie commande et une description du comportement de la partie opérative. Ceci est dû à l'utilisation de l'espace d'états du système pour la construction des modèles et par une caractérisation des ensembles d'états en points de vue pour le contrôleur.

Pour pallier le manque d'information concernant l'état du système lors des reconfigurations, les auteurs proposent l'ajout d'un *état non-significatif*. Cet état « *caractérise l'état du système par rapport à un mode donné quand les autres états de ce même mode ne le permettent pas* ». Autrement dit, quand aucun état d'un mode ne correspond à l'état réel du système, alors il est considéré que, dans ce mode, le système se trouve dans un état non significatif.

À travers ces travaux, les auteurs proposent une démarche de décomposition et de caractérisation d'un système en fonction de l'état des ressources. Cette démarche permet de représenter un système complexe par des modes de fonctionnements. De plus, l'utilisation des SyncCharts offre la possibilité de tester et d'implémenter les modèles construits par cette démarche. Cependant, il est toujours nécessaire d'effectuer des phases de validation, notamment sur le respect des spécifications, sur ces modèles. Ces phases ne sont pas incluses initialement dans la démarche proposée, mais ont fait l'objet de travaux plus récents [Ham04, Ham06, Ham09]. Néanmoins, la démarche dans son ensemble reste d'une grande complexité. Elle comporte de nombreuses étapes successives entre le début de la décomposition du système en points de vue jusqu'à l'obtention des modèles finaux dont les spécifications sont validées. Un expert de cette démarche est donc nécessaire pour l'utiliser. L'expertise demandée est d'autant plus grande que l'interprétation des modèles est difficile et que cette démarche est elle-même inclue dans une approche plus générale sur la sûreté de fonctionnement [Cra94].

2.7 Démarche de construction exhaustive des commutations

Des travaux conjoints entre les universités de Berkeley et de Philadelphie portent sur la synthèse de changement de mode pour atteindre des spécifications désirées [Koo01, Koo01a]. Leur objectif est de proposer une *démarche générique* de construction des modes et des commutations qui respecte les spécifications de chaque mode, même lors de commutations.

Pour les auteurs, un moyen naturel de réduire la complexité d'un système lors de la conception est de le décomposer en séquences de petits problèmes, plus facilement *interprétables*. Les auteurs considèrent qu'une décomposition en *modes de fonctionnement* d'un système est particulièrement adaptée pour garder une facilité d'interprétation du fonctionnement d'un système. Les auteurs définissent un *mode* comme étant « *l'action d'un système, sous l'effet d'un contrôleur connecté en boucle fermée* ». La difficulté est alors la capacité à commuter entre un nombre fini de modes représentant le système global. Chaque mode possède ses spécifications

propres et une commutation représente un état dans lequel le système est entre deux modes. Ces commutations peuvent être sources d'erreurs durant la conception car il est difficile pour le concepteur de prévoir tous les états intermédiaires possibles entre deux modes. Ainsi, déterminer des spécifications de commutation et les respecter permet d'éviter des défaillances ou des états indéterminés. De plus, les auteurs s'intéressent à la possibilité de définir une séquence finie de modes pour contrôler le système. Ceci est particulièrement adapté si le mode à atteindre n'est pas un mode directement atteignable mais ne l'est qu'en passant par un autre mode. Les auteurs déterminent les conditions de changement pour basculer entre modes et arriver au mode final. Les applicatifs pris sont des systèmes composés par un ensemble de composants indissociables et où seules les spécifications changent entre chaque mode.

Les auteurs considèrent que chaque mode, représentant un système sous l'action d'un contrôleur, possède un ensemble de trajectoires autorisées noté $\mathcal{R}_i$ où i est le mode considéré. Ainsi, une commutation d'un mode i vers un mode j n'est possible que si l'intersection des ensembles de trajectoires permises entre les deux modes est non nulle. Si tel est le cas, alors un contrôleur de commutation de modes est conçu spécialement pour commuter du mode i au mode j. Le contrôleur restreint le système pour que celui-ci atteigne l'ensemble des trajectoires autorisées dans les deux modes, ce qui rend ensuite la commutation effective. Dans le cas contraire [4], cela signifie qu'une commutation directe n'est pas possible et les auteurs recherchent une séquence de commutation permettant d'y parvenir.

Pour concevoir une démarche générique permettant de restreindre le comportement du système à un sous-ensemble de trajectoires autorisées dans les deux modes, les auteurs proposent une procédure en deux étapes. La première étape consiste à construire, dans chaque mode, des sous-modes représentant les trajectoires pour commuter vers le mode considéré ou de celui-ci vers d'autres modes. Chaque sous-mode représente l'intersection des ensembles de trajectoires des deux modes connectés et ces sous-ensembles de trajectoires sont autorisées par un contrôleur de commutation. Un contrôleur est donc construit par connexion entre mode. Avec cette architecture, les auteurs déterminent qu'il y a *au plus* $2N$ sous-modes à construire par modes en considérant N modes dans un système. Soit un total de $2N^2$ sous-modes.
La deuxième étape consiste à rechercher les commutations désirées entre modes. Dans les cas où des commutations directes sont identifiées comme impossibles, il est recherché l'existence d'une succession de contrôleurs restreignant à tour de rôle les trajectoires afin d'arriver au mode final désiré.

4. Celui où l'intersection des ensembles de trajectoires permises entre les deux modes est vide.

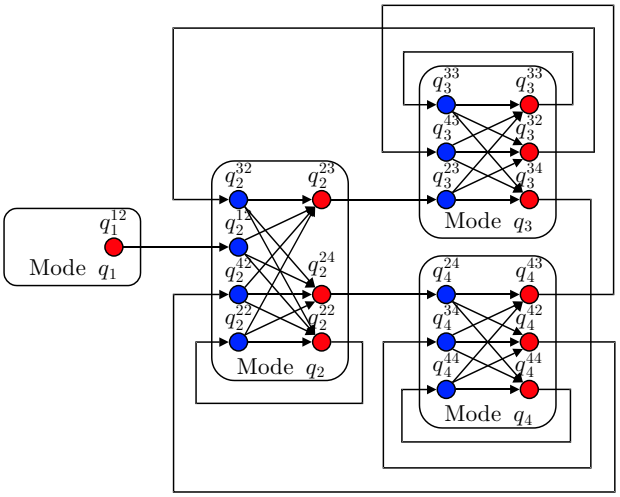

FIGURE 2.4 – Exemple de construction générique des commutations entre plusieurs modes

⌜ *Exemple*

La figure 2.4 illustre la proposition pour un système ayant 4 *modes. Ce système possède* 20 *sous-modes répartis entre ces modes. Cependant, le mode initial* q_1 *ne possède qu'un seul sous-mode servant à restreindre son ensemble de trajectoire afin de commuter vers le mode* q_2*. Tous les modes hormis* q_1 *sont connectés entre eux. Une commutation indirecte est, par exemple, de passer du mode* q_1 *au mode* q_3*. Cette trajectoire n'existant pas, une solution possible est de transiter par le mode* q_2*.* ⌟

En conclusion, les auteurs proposent une démarche générique pour la conception de modes de fonctionnements accompagnée d'une stratégie de commutation. Leur proposition s'articule autour de l'intégration de $2N$ sous-modes par mode pour un système à N modes. Le point fort de cette approche est de proposer une solution de complexité *polynomiale* pour résoudre la commutation entre modes. Les auteurs considèrent cette complexité raisonnable, permettant même de faire du calcul temps réel sur des systèmes constitués de plusieurs centaines de modes. De plus, la prise en compte de commutations possibles entre tous les modes permet d'éviter des erreurs de conception lors de l'étude des commutations. Néanmoins, leur approche ne suppose que des événements de commutation voulus et non des événements exigés[5] immédiatement comme pourraient l'être des événements de défaillances.

5. ou événements non désirés

2.8 Approche multi-modèle pour la sécurité de fonctionnement

Depuis de nombreuses années, le laboratoire Ampère travaille sur les S.A.P. dont les premiers travaux concernent la modélisation de systèmes multi-phase et la sûreté de fonctionnement [Rez93, Nie93, Rez95, Now96].

Les systèmes multi-phase sont des systèmes ayant des phases de fonctionnement distinctes. Dans chacune de ces phases, le système se comporte différemment selon les spécifications à respecter. Avec le temps, ces phases ont été définies comme des modes de fonctionnements où chacun est caractérisé par une configuration particulière qui doit respecter un ensemble de spécifications.

Le laboratoire Ampère, toujours dans le domaine de la sûreté de fonctionnement, orienta ses recherches vers des approches permettant de représenter un système à travers ses différentes configurations et qui assuraient que les spécifications attendues soient respectées, même lors de défaillances [Nie95, Kha99, Cha00, Kha02].

La suite de ce chapitre fait référence aux travaux menés qui ont précédé notre recherche. Ces travaux utilisent la Théorie de Contrôle par Supervision et adoptent une décomposition modale.

2.8.1 Sécurité opérationnelle avec un mode dégradé

Une des premières thèses réalisée au laboratoire Ampère dans ce domaine fut celle de Nourelfath [Nou97]. Ces travaux s'articulent autour de deux propositions. La première porte sur la difficulté d'implémenter une commande nominale issue de la S.C.T ; la deuxième est relative à la surveillance en vue d'assurer la continuité de service en présence d'une défaillance critique [Nou96a, Nou96b, Nie95, Nie96, Nou96, Nou04].

Dans sa première contribution, l'auteur s'intéresse à l'impossibilité d'implémenter directement les modèles fournis par la S.C.T. Il faut au préalable extraire une trajectoire du modèle ; cette trajectoire correspond alors à la loi de commande du système. Cependant, certaines trajectoires sont difficiles à implémenter à cause d'événements incontrôlables pouvant survenir à tout moment. L'auteur propose alors d'étendre la notion de contrôlabilité afin de pouvoir *forcer* un événement contrôlable devant un événement incontrôlable. Dans un même état où pourraient se produire deux événements, l'un contrôlable, l'autre incontrôlable, l'événement contrôlable forcé se produit *systématiquement avant* l'événement incontrôlable. Cela permet d'assurer que la trajectoire généré par l'occurrence de cet événement incontrôlable ne se produise. Physiquement, l'applicabilité de cette proposition est possible grâce à la rapidité de réaction des automates industriels. Avec cette nouvelle notion, des trajectoires qui étaient impossibles à implémenter à cause d'événements incontrôlables non voulus le deviennent.

Dans la deuxième contribution, l'auteur s'intéresse à la continuité de service lors de l'oc-

currence d'une défaillance. Dans l'étude, la survenue d'une seule défaillance à la fois est considérée. L'auteur propose d'inclure au modèle standard de la S.C.T. un module de surveillance chargé d'observer la présence d'une défaillance et d'assurer la continuité jusqu'à correction de celle-ci. La figure 2.5 illustre le modèle de la S.C.T augmenté du module de surveillance.

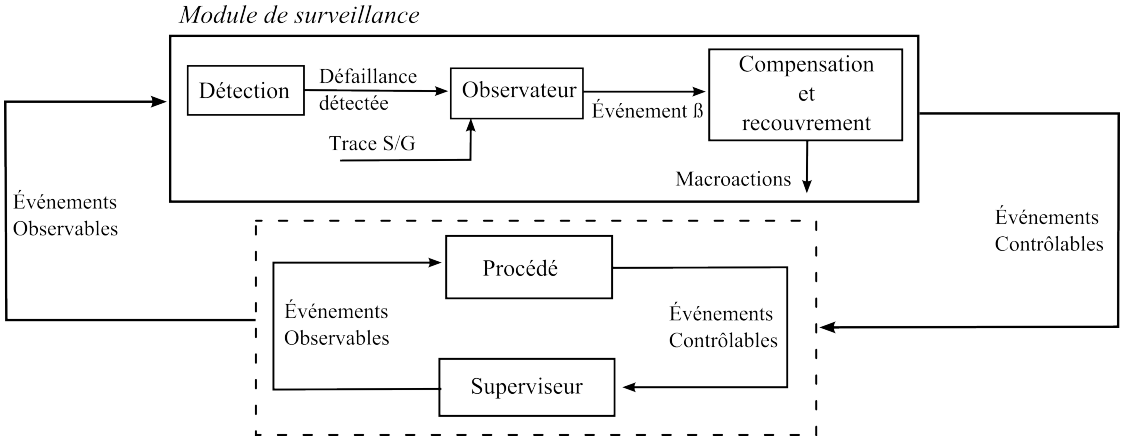

FIGURE 2.5 – Module de surveillance ajouté au modèle de la S.C.T

Le module de surveillance comporte trois fonctions : détection, observation et compensation/recouvrement. En l'absence de défaillance, le module de surveillance n'a pas d'influence sur la commande ; le module est inactif. La première fonction du module de surveillance est la détection d'une défaillance. Lorsque tel est le cas, un signal est envoyé à la fonction d'observation qui détermine le type de compensation à utiliser grâce à un historique du comportement du système. Le type de compensation est envoyé par l'événement β à la fonction de compensation/recouvrement. celle-ci agit par des actions globales, dites macroactions, sur le système. Ces actions ont pour but de maintenir la continuité de service dans un état *dégradé* du système. Lorsque l'élément défaillant est réparé, la fonction recouvrement laisse le système fonctionner en mode nominal et remet le module de surveillance en veille jusqu'à la prochaine défaillance.

Afin de ne pas créer de conflit avec deux superviseurs différents, le superviseur nominal doit être désactivé lorsque survient une défaillance. Le système est libéré des contraintes exercées par ce superviseur mais se retrouve restreint au comportement admis par le superviseur du module de surveillance. Lorsque la défaillance est corrigée, le superviseur du module de surveillance libère le système afin que le superviseur nominal puisse exercer à nouveau ses contraintes. Il y a donc un enchaînement d'activation de superviseurs (nominal et dégradé) selon l'occurrence d'une défaillance. À aucun moment les deux superviseurs ne doivent exercer leurs contraintes simultanément au risque de bloquer le système. Pour réaliser cela, les auteurs proposent l'ajout d'un *état puits* à chaque superviseur. Dans cet état, le superviseur n'exerce aucun contrôle et y reste jusqu'à la survenue d'un événement particulier. Pour le superviseur nominal, cet événement particulier est celui représentant une défaillance, alors que pour le superviseur du module

de surveillance, l'état particulier est celui fourni par l'observateur et est dépendant de la trace du système.

En conclusion, ces travaux permettent de décrire un système à travers deux configurations particulières (nominale et dégradée) ; chacune de ces configurations ayant un ensemble de spécifications à respecter. L'utilisation de la S.C.T. pour le mode nominal assure que les spécifications sont respectées, et l'utilisation d'un module de surveillance permet d'assurer une continuité de service quel que soit l'état de défaillance du système. Cependant, ces travaux ne se limitent encore qu'à deux configurations et à l'occurrence d'une seule défaillance à la fois. De plus, seul le mode nominal est celui étudié par la S.C.T : rien ne garantit que la commande de la deuxième configuration respecte bien totalement les spécifications voulues.

2.8.2 Approche multi-modèle

Dans la continuité des travaux précédents, les travaux de thèse de Kamach [Kam04] ont pour but de proposer une démarche, appelée multi-modèle, permettant de représenter un système à n modes de fonctionnements et où plusieurs défaillances peuvent survenir [Kam05b, Kam05a, Kam06a, Kam06b].

Les difficultés résident dans l'enchaînement d'activations des superviseurs et dans la détermination des états du système lors de ces activations. En effet, les travaux précédant stipulaient que la compensation d'un système [6] lors d'une défaillance dépendait de la trace [7] de celui-ci. Avec un système à n modes, il est donc nécessaire de réaliser un suivi de trajectoires indépendamment des modes activés. De plus, il est également nécessaire de rester dans le cadre mathématique de la S.C.T. afin de continuer à assurer le respect des spécifications dans chaque mode.

Dans ses propositions, l'auteur considère qu'un seul superviseur est actif à la fois ; la désactivation est représentée par l'ajout d'un état puits. Cette hypothèse d'unicité de mode actif est la même que celle considérée dans d'autres travaux [ADE81, Bal00, Dan00, Jah94, Koo01, Mos00]. Elle a pour but d'éviter l'incohérence et l'indéterminisme provoquées par l'existence de plusieurs contrôleurs commandant globalement le même système. L'auteur considère également que l'ensemble des événements représentant les défaillances du système est disjoint de l'ensemble des événements communs. Ainsi, il est possible de séparer l'étude du système lors de la conception en deux parties. La première est la création des modèles représentant le comportement interne des modes ; la deuxième est une extension des modèles construits dans la première étape. Cette extension ajoute le comportement commutatif du système. La démarche globale proposée par l'auteur est illustrée figure 2.6 et a pour but d'aider le concepteur dans la construction de la loi de commande.

6. C.-à-d. le maintien de la continuité de service dans un fonctionnement dégradé.
7. Une trace correspond aux différents événements qui se sont précédemment produits et qui ont conduit à l'état actuel.

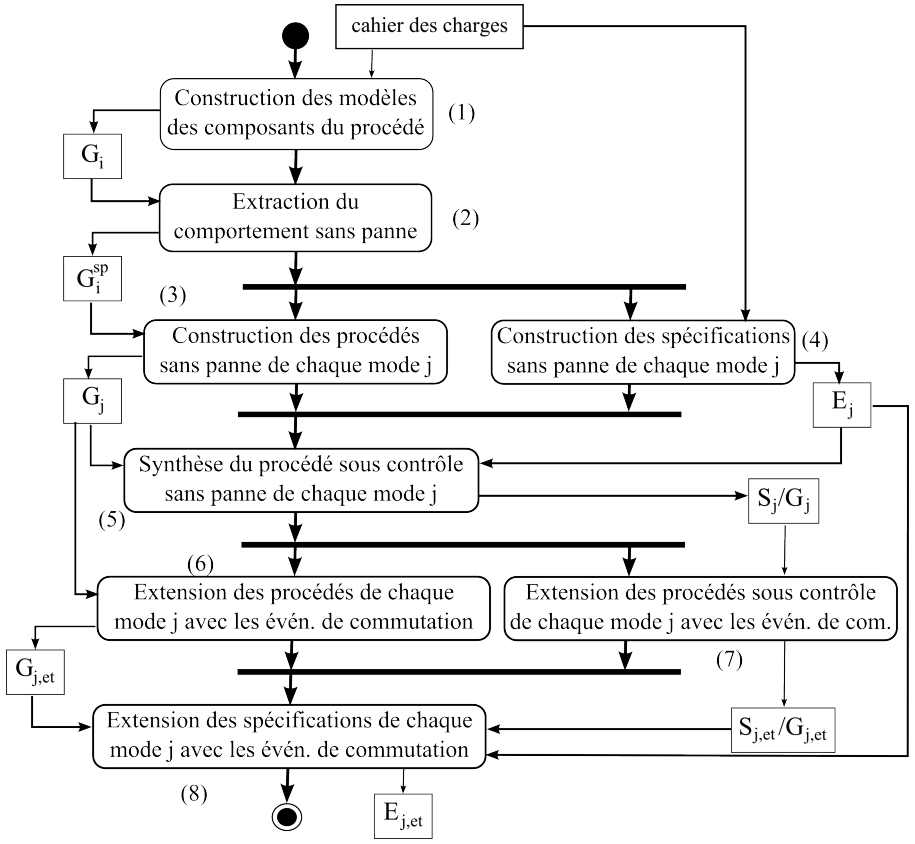

FIGURE 2.6 – Démarche proposée pour l'approche multi-modèle

À partir du cahier des charges, le concepteur construit les modèles des composants G_i (étape 1 sur la figure 2.6). Le concepteur extrait ensuite de ces modèles leur comportement sans défaillance, appelé G_i^{sp} (étape 2). Dans ces modèles, les événements de défaillance ne sont pas pris en compte et toutes les trajectoires ayant de tels événements sont supprimées. À partir des composants utilisés dans le mode j, le concepteur construit les modèles des procédés G_j par composition parallèle des modèles des composants G_i (étape 3). Le concepteur est également en mesure de construire les modèles de spécifications E_j représentant les contraintes à respecter dans le mode j (étape 4). Il effectue ensuite la synthèse des procédés G_j et des spécifications E_j afin d'obtenir les modèles des procédés sous contrôle S_j/G_j (étape 5). Ces modèles de procédés sous contrôle représentent le comportement admissible du système dans chaque mode. Cependant, les événements de commutation sont manquants. L'ajout de ces événements se fait par une phase d'extension des modèles de modes G_j afin d'y inclure le comportement commutatif du système (étape 6). Ces modèles étendus se nomment $G_{j,et}$ et représentent le comportement de chaque mode étendu avec les événements de commutation. L'inactivation liée au changement de mode se fait par l'ajout, lors de la phase d'extension, d'un état puits appelé $q_{j,in}$ qui représente le mode dans un état inactif. L'ajout des événements de commutation sur les modèles des procédés sous contrôle S_j/G_j se fait également par une phase d'extension en analysant cette fois le

comportement des modèles précédemment étendus $G_{j,et}$ (étape 7). Le concepteur obtient alors les contrôleurs de mode $S_{j,et}/G_{j,et}$ qui représentent les comportements internes et commutatifs admissibles pour le système. Enfin, pour obtenir les modèles $E_{j,et}$ représentant les modèles des spécifications étendus, Il procède une dernière fois à une phase d'extension des modèles de spécification E_j à partir des informations des modèles $G_{j,et}$ et $S_{j,et}/G_{j,et}$ (étape 8).

Pour résoudre les difficultés rencontrées et valider la démarche proposée, l'auteur propose une fonction de *suivi de trajectoire*. Cette fonction identifie les états de départ et d'arrivée dans chaque mode rendant ainsi les modèles déterministes par une labélisation des événements. L'auteur démontre qu'un seul superviseur par mode est suffisant pour respecter les spécifications indépendamment des états de départ et d'arrivée. Cette démonstration s'appuie sur la possibilité d'identifier les états de départ et d'arrivée grâce au suivi de trajectoire. Pour finir, l'auteur démontre que les modèles des procédés sous contrôle étendus sont contrôlables et non bloquants. Pour vérifier la contrôlabilité, l'auteur part de l'hypothèse que les modèles non étendus sont déjà par définition contrôlables et non-bloquants. Comme les événements de commutation ajoutés lors de la phase d'extension sont toujours des événements autorisés, alors les procédés sous contrôle étendus restent forcément contrôlables. Pour la vérification du non blocage, l'auteur démontre que les contrôleurs étendus sont accessibles et co-accessibles, quel que soit l'état de départ.

En conclusion, la démarche proposée permet de concevoir un système quel que soit le nombre de modes de fonctionnement. L'utilisation de la S.C.T. assure également le respect des spécifications. Cependant, cette démarche n'est pas complète. En effet, l'auteur ne dit pas explicitement quels sont les modèles de base nécessaires pour utiliser la démarche : est-ce les modèles des composants incluant les événements de commutation ou est-ce ceux ne les incluant pas ? Enfin, la contrôlabilité sur les modèles représentant les procédés sous contrôle étendus n'est pas vérifiée. Cela implique qu'il n'est plus possible de garantir le respect des spécifications sur ces modèles finaux.

Dans les exemples présentés, l'auteur illustre systématiquement les modèles des composants incluant les événements de commutation, mais n'explicite pas comment obtenir les modèles sans panne (étape 2 de la figure 2.6). Ces modèles sont construits manuellement et demandent donc un expert pour être modifiés. De plus, les phases d'extension, celles où le comportement commutatif est rajouté (étape $6, 7$ et 8), ne sont pas clairement définies. L'auteur fait référence plusieurs fois à l'ajout d'événements de commutation là où *ils peuvent être générés* mais aucune indication sur la façon de détecter les états où ils peuvent se produire n'est donnée. Si l'idée guidant l'extension est compréhensible sur un exemple très simple, celle-ci peut devenir très complexe, voire inapplicable dans certains cas où tous les événements de commutation seraient incontrôlables. Ces phases d'extensions ont donc besoin d'un expert pour être mises en œuvre. Enfin, un des derniers problèmes soulevés concerne directement les événements de commutation.

2.9 Synthèse

Dans ce chapitre traitant de la problématique de la gestion de mode, nous avons présenté plusieurs travaux proposant des solutions différentes. Nous proposons, pour chacun des travaux et pour plus de clarté, un tableau de synthèse 2.1 pour regrouper les points importants mentionnés en début de chapitre.

Définition d'un mode Chaque travail propose une définition pour un mode. Bien que différentes sur la forme, ces définitions contiennent des idées communes perceptibles sur le fond. Il ressort en effet qu'un mode, dans un système, est une abstraction, un concept, servant à réduire la complexité et améliorer l'interprétation qui peut être faite sur un système. Cette réduction et cette amélioration résulte d'un découpage de l'espace d'états du système tel qu'il représente un fonctionnement temporaire de celui-ci. Un mode est par conséquent déduit des contraintes, des composants utilisés, ou d'une qualité de service particulière que l'on attend du système. Ainsi, la modification d'une contrainte, une tâche différente à accomplir, un changement de composants, une meilleure qualité de service, etc. sont autant de raisons qui justifient un changement de mode du système. L'ensemble de tous ces critères correspond à une configuration particulière du système où le moindre changement d'un de ses critères revient à un changement de configuration.

Propositions et points faibles Comme dit à plusieurs reprises, une représentation modale est adaptée et courante en industrie pour décomposer un système complexe et le décrire. Depuis le GEMMA, toutes les propositions se concentrent sur une démarche permettant d'aider dans la conception des modes de fonctionnement suivant différents critères tels que spécifications, objectifs, ressources, etc. De plus, il est aussi fait références à des étapes de vérification sur les commutations entre modes. Cependant, peu sont complètement définies, automatisables et prennent en compte tous les modes et toutes les évolutions possibles du système. La conception et la gestion des modes est un domaine complexe car une démarche trop générale rend inapplicable la proposition. À l'opposé, une démarche trop détaillée n'est plus générique et ne s'applique qu'à un type précis de système. Enfin, une conception mal définie implique des erreurs pouvant y survenir. Il est alors nécessaire de considérer tous ces problèmes à la fois pour éviter les problèmes de conception.

TABLE 2.1 – Tableau de synthèse des travaux bibliographiques

	Mode (définition)	Proposition	Points faibles	Langages, méthodes, théories	Type de commutations	Unicité d'activité de mode	Exhaustivité de modes	Exhaustivité de commutations	Proposition mathématiquement définie	Type de Description du comportement
GEMMA	Appartient à une famille et représente un sous-ensemble dans lequel est décrit une loi de commande locale.	Canevas de modes et de commutations.	Démarche mathématiquement non définie et ensemble de modes prédéfini	Canevas graphique	Conditions logiques	Oui	Non	Non	Non	Partie commande
Mode-Charts	Partition de l'espace d'états d'un système	Démarche de connexion des modes.	Difficilement applicable	State-charts	Événement et conditions logiques	Non	Oui	Non	Oui	Partie commande
Mode-automata	Ensemble d'états dans lequel le système se trouve avant de commuter vers un état ne faisant pas parti de cet ensemble. En terme d'implémentation, un mode est une fonction.	Démarche de conception et de vérification.	Perte d'informations possible en changeant de modes.	Automate + Lustre	Conditions logiques	Oui	Non	Non	Oui	Partie commande
Spécifications robustes	Représente une fonction, fournit une qualité de service.	Démarche de conception, de vérification et d'implémentation de modules d'électroniques embarqués	Explosion combinatoire	Logique combinatoire	Conditions logiques	Oui	Oui	Oui, par des étapes de recherche	Non	Parties commande et opérative

Continue sur la page suivante

Tab. *2.1* – Tableau de synthèse des travaux bibliographiques (suite)

	Mode (définition)	Proposition	Points faibles	Langages, méthodes, théories	Type de commutations	Unicité d'activité de mode	Exhaustivité de modes	Exhaustivité de commutations	Proposition mathématiquement définie	Type de Description du comportement
CASPAIM	Ensemble d'états caractérisant une ressource ou un ensemble de ressources selon un point de vue (PdV). Le PdV dépend de l'opérateur qui utilise les ressources.	Démarche de conception pour les systèmes complexes	Nombreuses phases manuelles exigeant un expert	SyncCharts	Événements et Conditions logiques	Oui	Oui	Oui, mais fait manuellement	Non pour la conception, Oui pour la vérification	Parties opérative et commande
Construction exhaustive des commutations	Opération sur le système de contrôle qui garantit la sélection d'un sous-ensemble particulier (répondant aux contraintes du modes)	Démarche générique et automatisable de changement de contraintes indépendamment du nombre de modes	Commutations demandées sur événements contrôlables	Langage informatique	Événements	Oui	Oui	Oui, étape de conception.	Non	Partie commande
Sécurité opérationnelle	Représente le comportement d'un système sous des contraintes particulières (propres aux modes)	Démarche de changement de contraintes pour un mode dégradé	Deux uniques modes possibles	S.C.T.	Événements	Oui	Non	Non	Oui, uniquement pour le mode nominal	Partie opérative (mode nominal) et partie commande (mode dégradé).
Approche Multi-modèle	Fixe les moyens ou les ressources nécessaires à l'exécution d'une tâche prescrite.	Démarche de conception pour un système à n modes	Construction des commutations manuelle et contrôlabilité des modèles finaux non vérifiées	S.C.T.	Événements	Oui	Oui	Non, définies manuellement	Pas complètement	Partie opérative (comportement interne), partie commande (comportement commutatif).

Unicité d'activité de mode Lorsqu'un système est décrit par ses différentes modes de fonctionnement, il peut se produire une incohérence de fonctionnement dans le cas où *plus d'un mode est actif en même temps*. Les travaux sur les mode-charts [Jah94] ont mis en évidence ce cas lorsque deux modes sont connectés en parallèle. Pour pallier ce problème, la plupart des travaux présentés dans ce chapitre connectent les modes de manière à avoir une activation séquentielle de ceux-ci [ADE81, Mar98, Koo01]. Dans le cas où les modes sont représentés par des modèles distincts [8], il convient d'ajouter un état puits [Cha00, Ham04, Kam05b] afin d'éviter que deux lois de commandes soient actives en même temps. Ainsi, l'ajout de cet état inactif permet d'assurer qu'un seul mode est actif à chaque instant.

Exhaustivité du nombre de modes et des commutations Un des grands problèmes dans la conception d'un système par une représentation modale est celui de la détermination des modes possibles. Oublier de représenter un mode où le système peut potentiellement se retrouver est équivalent à ne plus avoir d'état qui corresponde à l'état physique du système. Ce cas peut potentiellement conduire à un blocage du système ou, pire, à des dommages physiques. Pour éviter cette situation, plusieurs propositions ont été faites. Globalement, il est possible de séparer ces propositions selon trois solutions. La première est celle de la création d'un canevas des modes possibles et des commutations existant entre eux, comme dans le GEMMA ou dans la démarche CASPAIM. Cette proposition fixe *a priori* tous les modes et les commutations qui peuvent exister. Cependant, à cause de la complexité croissante des systèmes, il n'est plus possible de déterminer à l'avance tous les modes qu'un système peut avoir, et encore moins les commutations entre eux. Cela reviendrait à être capable de déterminer tous les besoins possibles de tous les systèmes imaginables, mais également toutes les situations non idéales ou non désirées comme des pannes, une cadence de production réduite, une production maximale forcée malgré la présence d'une panne, etc. En bref, un système peut avoir une infinité de modes selon les besoins. La deuxième solution, utilisée dans les travaux de l'IRCCyN et dans les travaux des universités de Berkeley et de Philadelphie, consiste à créer lors de la conception tous les modes qui peuvent exister dans le système par une exploration de l'espace d'états. Cela est possible pour les systèmes électroniques binaires qui ont un espace d'états limité. Cependant, même s'il est borné, le nombre de modes est évidemment conséquent et le problème d'explosion combinatoire reste présent. Il reste alors la troisième voie, prise par les travaux sur l'approche multi-modèle et également par l'IRCCyN pour réduire les problèmes de la deuxième solution, qui consiste à ne concevoir que les modes strictement nécessaires pour un système donné en recherchant les commutations possibles entre modes afin de déterminer si le système peut être amené à commuter vers un mode non prévu. Cette solution permet de ne pas être limité à un canevas pré-défini tout en limitant le nombre de modes et donc l'explosion combinatoire.

8. Comme par exemple un modèle par mode.

Démarche mathématiquement définie Depuis la proposition du GEMMA, tous les travaux ont tenté de définir plus rigoureusement les étapes de construction des modèles. Les systèmes devenant de plus en plus complexes, ces étapes de construction le deviennent également et il est nécessaire pour respecter des propriétés sur ces modèles d'être en mesure de les définir complètement et mathématiquement. Pour arriver à cela, certains travaux se sont orientés vers des formalismes facilitant les étapes de vérification qui démontrent que les modèles obtenus respectent bien les spécifications attendues [McM92]. Cependant, ces étapes se réalisent après les étapes de conception, ce qui augmente le temps de développement. Désirant avoir au plus tôt des modèles sûrs par construction, des travaux se sont basés sur des démarches qui permettent de construire des modèles respectant de manière sûres les spécifications . Il est alors inutile de vérifier par la suite le respect de ces contraintes [Ram89]. En conséquence, la conception d'un système par une approche modale demande des démarches sûres et mathématiquement définies afin d'être en mesure d'utiliser une telle approche dans un milieu industriel exigeant sur les propriétés à respecter.

Type de description Il a été discuté à plusieurs reprises de la description du comportement adoptée dans les différentes propositions présentées. Cette description du comportement fait référence soit à la partie commande, soit à la partie opérative. En effet, lors de la conception d'un système par ses modes de fonctionnement, il existe plusieurs stratégies pour atteindre les objectifs fixés. Une stratégie consiste à débuter la conception par une description détaillée du comportement de la partie opérative. Cette description concerne la construction des modèles sur la base des états d'un système opératif. De ces états sont ensuite créés les modes et leurs commutations. Une autre stratégie consiste à décrire le comportement de la partie commande, celle qui est appliquée aux systèmes. Ce comportement ne se caractérise pas suivant les états possibles du système opératif mais sur un comportement attendu du système. La première stratégie a l'inconvénient d'être soumise à l'explosion combinatoire contrairement à la deuxième. En contre-partie, la première assure de n'oublier aucun état ou comportement possible du système alors que la deuxième ne prend en compte qu'une partie seulement de l'ensemble d'états qui implique par conséquent des oublis possibles lors de la conception. Le choix de la stratégie pour la conception est cruciale car implique directement les problèmes qui surviendront par la suite, notamment lors de l'étude des commutations entre modes. Si la deuxième stratégie a peu de problème pour les étudier, du fait d'avoir décrit au préalable le comportement attendu de la partie commande, ce n'est pas le cas de la stratégie qui consiste à décrire un système par tous les états et toutes les transitions possibles car l'explosion combinatoire rend le nombre de commutations extrêmement grand. Le choix pris par certains travaux est d'utiliser les deux descriptions conjointement et de les faire correspondre pour éviter toute incohérence. Cependant, cette correspondance est difficile à mettre en place et demande un expert. Cependant, c'est l'unique moyen pour éviter une perte d'information dans les modèles.

2.10 Conclusion

Les démarches exposées ont pour but général d'aider à la conception de lois de commande par une approche modale en s'intéressant aussi bien à la construction des modèles de modes qu'aux connexions entre ceux-ci. Concernant la construction de modèles, plusieurs travaux proposent soit une liste exhaustive de modes prédéfinis[ADE81, Dan02], soit une procédure où tous les modes sont recherchés et créés [Koo01, Rak05]. Cette recherche assure que le système ne puisse commuter vers un mode non prévu par le concepteur. Cependant, prendre en compte tous les modes et commutations possibles provoque une explosion combinatoire. Imposer un canevas prédéfini de modes et commutations en revanche permet d'éviter le problème de l'explosion combinatoire mais ne permet pas de générer toutes les commutations possibles. Ces raisons nous poussent à laisser le concepteur libre du choix des modes à créer, tout en respectant les données du cahier des charges. Cependant, il est important d'étudier les commutations entre chaque mode, et pas seulement se focaliser sur la construction des modèles de modes, comme réalisée dans certains travaux [Koo01, Dan02, Rak05]. L'étude des commutations permet d'éviter la perte d'informations relevée dans les travaux sur les mode-automata [Mar92, Mar98]. De plus, le nombre de commutations entre modes peut être très élevé. Il est donc important de disposer d'une démarche formelle et générique, potentiellement automatique, pour l'étude de ces commutations.

Nos travaux sont dans la continuité de ceux précédemment réalisés au laboratoire Ampère qui portent sur la conception de modèles représentant un comportement modal dans un contexte de gestion de mode [Cha00, Nou04, Kam05b]. Ces travaux se basent sur la Théorie de Contrôle par Supervision qui assure que les modèles créés sont sûrs par construction qui permet de supprimer les phases de vérification a posteriori. Ces travaux proposent l'ajout d'un état inactif pour activer ou désactiver les contrôleurs de mode, de séparer les études concernant les comportements internes et externes aux modes, et de réaliser le suivi des traces qui mènent à un événement de commutation afin d'éviter la perte d'information liée à un changement de mode. Les derniers travaux précédent les nôtres ont également montré la suffisance d'un superviseur par mode pour gérer toutes les commutations possibles. Cependant, ces travaux n'ont pas complètement formalisé la démarche de construction des modèles de mode, plusieurs étapes sont réalisées manuellement, sans procédure pour les détailler. Enfin, les commutations sont ajoutées manuellement dans les modèles. Ceci pose un problème de contrôlabilité, lié à une représentation des commutations non exhaustive dans les modèles, ce qui peut mener le système dans un état non prévu par le concepteur.

À la lecture des précédents sections et chapitres, nous voulons que notre proposition, en utilisant la théorie de contrôle par supervision, soit une démarche complètement définie pour la construction des modèles de mode. Elle doit également garder des travaux précédent la séparation des différentes études suivant les comportements générés. Plus particulièrement, notre

proposition doit proposer une démarche qui permette au concepteur de spécifier des contraintes sur les commutations du système tout en assurant une contrôlabilité sur celui-ci. La démarche doit permettre d'obtenir un modèle par mode où peut être extraite la loi de commande à implémenter dans le système physique.

3

Démarche de conception

Sommaire

3.1 Introduction

Dans les chapitres précédents, nous avons présenté la théorie des langages et la théorie de contrôle par supervision. Nous avons également présenté les principales contributions s'inscrivant dans la problématique de gestion de modes. Les verrous scientifiques qui subsistent relèvent de la définition des modes et de leurs commutations. Les difficultés majeurs sont de respecter les spécifications liées aux modes et aux commutations. À ce titre, la plupart des travaux à notre connaissance font référence à des études de validation *a posteriori*. Ceci contraint à reprendre toute l'étude si les spécifications ne sont pas validées. Nous avons ainsi retenu de définir une démarche formelle de construction de modes et de commutations de mode permettant en amont d'assurer le respect des spécifications attendues. Dans ce chapitre nous présentons dans un premier temps la démarche de modélisation des composants et des modes, et dans un second temps les différentes étapes de la démarche assurant le respect des spécifications [Far09a].

Ce chapitre se découpe ainsi en huit parties. La première présente l'exemple directeur traité tout au long de ce chapitre. La deuxième partie détaille pas à pas la mise en œuvre de la démarche. Les cinq parties suivantes sont les différentes étapes de cette démarche. Enfin, nous proposons une comparaison avec l'approche centralisée, exposée dans le chapitre 1. Nous conclurons ce chapitre sur les apports de cette démarche au regard de nos objectifs.

3.2 Exemple directeur

3.2.1 Description du système

Pour faciliter la compréhension de la démarche proposée, nous allons l'illustrer par un exemple. Cet exemple a été choisi pour permettre une représentation des modèles obtenus tout au long des étapes de conception. Les modèles restent donc relativement simples et faciles à interpréter. Le système support, illustré figure 3.1, est un système de production constitué de quatre machines de production C_1, C_2, C_3, C_4 et d'un stock interne B. Les stocks amont et aval à ce système ne sont pas considérés. Le cahier des charges précise que ce système peut être utilisé dans deux modes de fonctionnement, un mode nominal et un mode dégradé.

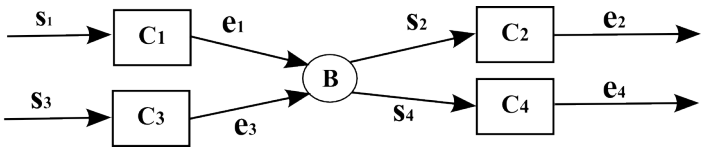

FIGURE 3.1 – Structure du système de production

Dans ce système, les machines C_1 et C_3 reçoivent des pièces de l'extérieur. L'arrivée d'une pièce, qui correspond au début de la tâche, est symbolisée par l'occurrence des événements s_1

pour la machine C_1 et s_3 pour la machine C_3. La fin de tâche est symbolisée par l'occurrence de l'événement e_1 pour la machine C_1 et e_3 pour la machine C_3. Les fins de tâche, représentant la sortie d'une pièce de C_1 ou de C_3, incrémentent le stock B d'une unité. Lorsqu'une pièce est présente dans le stock, une tâche peut être lancée sur la machine C_2 (par l'occurrence de l'événement s_2) ou sur la machine C_4 (par l'occurrence de l'événement s_4). Le lancement d'une tâche sur la machine C_2 ou C_4 décrémente le stock B d'une unité. La fin de tâche est symbolisée par les événements e_2, pour la machine C_2, et e_4, pour la machine C_4.

3.2.2 Décomposition modale

Nous avons décrit le système physique et les relations entre composants. Nous souhaitons maintenant décrire les différents modes du système.

Le système possède un mode *nominal* dans lequel la machine C_3 ne doit pas produire. La machine C_1 produit des pièces, une par une, qui sont placées dans le stock B. Ces pièces sont ensuite consommées par les machine C_2 ou C_4, sans notion de priorité, qui produisent chacune des pièces. La machine C_1 peut tomber en panne durant le déroulement d'une tâche. Cette panne est symbolisée par l'événement f_1. L'occurrence de l'événement f_1 implique une commutation du système du mode *nominal* vers un mode *dégradé* dans lequel la machine C_1, en panne, n'est plus utilisée jusqu'à l'occurrence de l'événement r_1 qui implique le retour du système dans le mode *nominal*.

Le système possède par conséquent un mode *dégradé*. Ce mode représente le comportement que nous désirons lorsque la machine C_1 est en état de panne. Dans ce mode, La machine C_1 est remplacée par la machine C_3 – la machine C_3 est dite redondante avec la machine C_1. Cependant, la machine C_3 est moins rapide que la machine C_1, il n'est dont plus nécessaire d'utiliser deux machines (C_2 et C_4) pour consommer les pièces du stock B. De ce fait, seule la machine C_2 reste active pour produire. La machine C_4 est donc stoppée pendant ce laps de temps où le système est dans le mode *dégradé*.

3.2.3 Modèle des composants

Le système possède quatre machines et un stock. Chaque machine est considérée ici comme un composant. Le modèle de chaque composant est illustré sur la figure 3.2 et est défini par la définition 23. Le stock n'est pas considéré comme un composant mais plutôt comme une spécification du système. Ceci correspond à un choix classique de la Théorie de Contrôle par Supervision [Won09] car le stock n'est générateur d'aucun événement qui lui est propre. Il n'est là que pour représenter une relation de contrainte entre les composants en mettant leurs événements en relation.

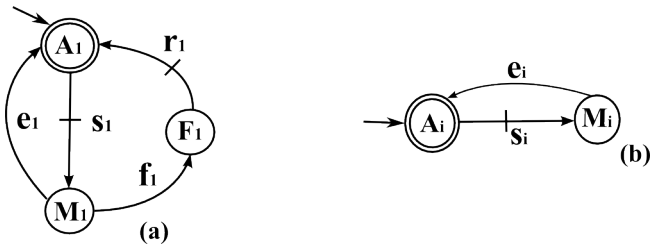

FIGURE 3.2 – (a) modèle de la machine C_1 – (b) modèle des machines C_2, C_3 et C_4

3.3 Présentation de la démarche

Notre approche, illustrée figure 3.3 sous forme d'un diagramme d'activités UML, repose sur l'utilisation de la Théorie de Contrôle par Supervision et sur une démarche progressive qui propose plusieurs étapes de construction des modèles. Notre démarche comporte cinq étapes de construction issues de nos travaux [Far09c, Far09b, Far08]. La première regroupe la construction des différents modèles de composants et de spécifications (section 3.4). L'étape 2 concerne l'étude du comportement du procédé sous contrôle dans chaque mode, sans tenir compte des commutations de mode (section 3.5). Dans l'étape 3, un enrichissement des modèles avec les composants et les spécifications de commutation est réalisé (section 3.6), ce qui permet ensuite dans l'étape 4 d'étudier des occurrences des événements de commutation, et de vérifier que les changements de modes sont possibles (section 3.7). Enfin, dans la dernière étape, les modèles obtenus sont réduits et font apparaître un état représentant l'inactivité du mode (section 3.8).

Du fait de l'utilisation de la Théorie de Contrôle par Supervision, ces étapes ne peuvent être mises en œuvre que par un spécialiste. Pour aider ce concepteur à construire des modèles répondant aux cahier des charges, chaque étape du processus est suivie d'une étape de validation (sections 3.4.2, 3.5.4, 3.6.4 et 3.7.3). La modélisation au plus tôt des contraintes imposées au système (par les spécifications) ainsi que les étapes de validation permettent au concepteur de faire régulièrement valider son travail par le client, l'aidant ainsi à répondre au mieux à ses besoins.

3.4 Formalisation du cahier des charges

3.4.1 Construction des modèles

Un système est constitué d'un ensemble de composants physiques. Certains de ces composants sont reliés entre eux afin de créer des fonctions spécifiques. Ces composants peuvent tomber en panne ou être activés suite à une tâche à réaliser. Ceci amène à caractériser un *ensemble de composants* utilisé par le système.

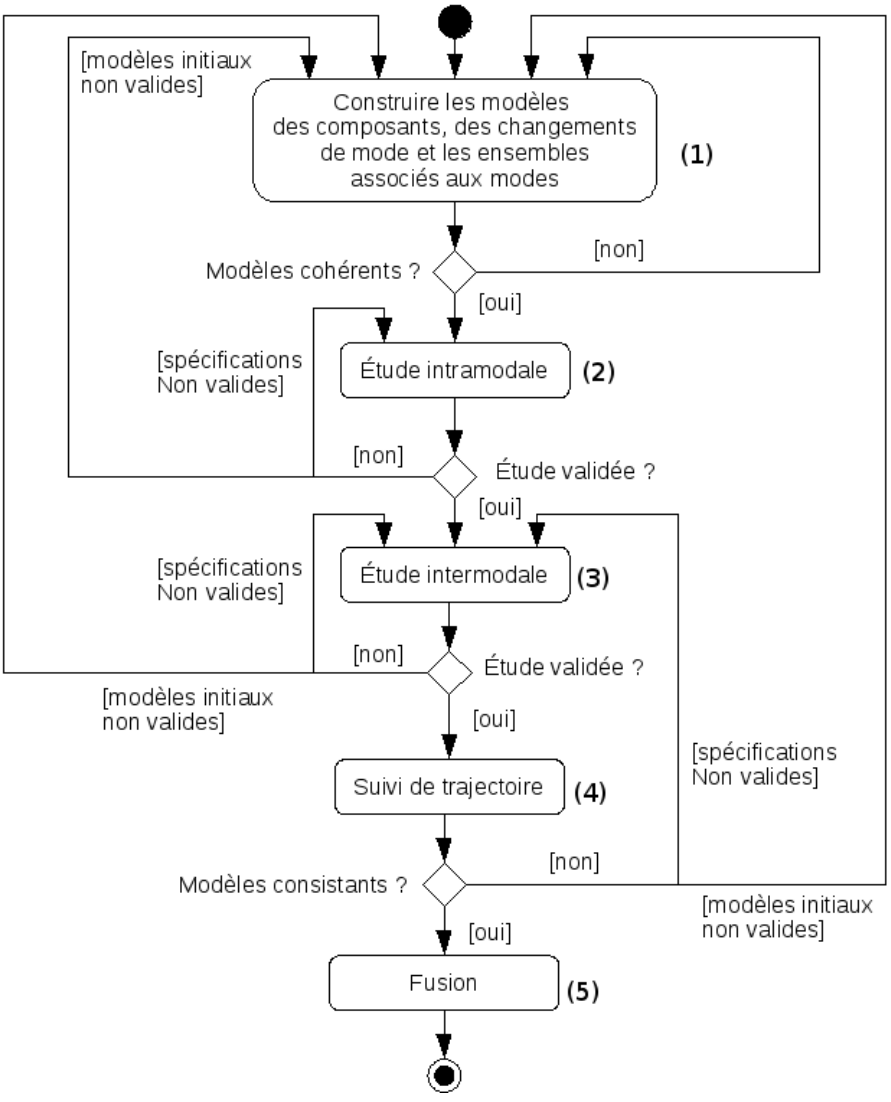

FIGURE 3.3 – Démarche de modélisation proposée

Définition 22 (Ensemble de composants)
L'ensemble des composants est appelé $\mathcal{C} = \{C_1, C_2, \ldots, C_i\}$, *où* $i \in \mathbb{N}$ *et* $i \geq 1$. ♦

Dans notre exemple, $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$: les composants sont des machines de production. Chaque composant est modélisé en suivant la définition suivante :

Définition 23 (Modèle d'un composant)
Un composant C_i *est modélisé par un automate* $G^{C_i} = (Q^{C_i}, \Sigma^{C_i}, \delta^{C_i}, q_0^{C_i}, Q_m^{C_i})$, *avec :*

- Q^{C_i}, $Q_m^{C_i}$ *et* $q_0^{C_i}$ *sont respectivement l'ensemble des états, l'ensemble des états marqués et l'état initial du composant* C_i ;
- Σ^{C_i} *est l'ensemble des événements du composant* C_i, *et inclut quatre sous-ensembles :*
  - $\Sigma^{C_i} = \Sigma_c^{C_i} \cup \Sigma_{uc}^{C_i}$ *avec* $\Sigma_c^{C_i} \cap \Sigma_{uc}^{C_i} = \emptyset$. Σ_c *et* Σ_{uc} *sont respectivement les sous-ensembles disjoints des événements contrôlables et incontrôlables du composant* C_i ;
  - $\Sigma^{C_i} = \Sigma_{\rightleftarrows}^{C_i} \cup \Sigma_{\circlearrowleft}^{C_i}$ *avec* $\Sigma_{\rightleftarrows}^{C_i} \cap \Sigma_{\circlearrowleft}^{C_i} = \emptyset$. $\Sigma_{\rightleftarrows}^{C_i}$ *est l'ensemble des événements de commutation de* C_i. $\Sigma_{\circlearrowleft}^{C_i}$ *sont les autres événements du composant.*
- δ^{C_i} *est la fonction de transition. Elle inclut* $\delta_{\rightleftarrows}^{C_i}$ *qui représente la fonction de transition de commutation. Une transition de commutation est une transition dans laquelle l'événement est inclus dans* $\Sigma_{\rightleftarrows}^{C_i}$. ♦

À partir des définitions 22 et 23, il est possible de définir un ensemble de modes représentant les différents comportements admissibles de notre système.

Définition 24 (Ensemble de modes)
Un ensemble de modes est défini par $\mathcal{M} = \{M_0, M_1, \ldots, M_j\}$, *où* M_j *est le nom du mode représenté.*

Cet ensemble représente tous les modes du système.

Remarque 2
Le nom d'un mode est défini par le concepteur ou le cahier des charges et n'est là que pour améliorer l'interprétation des modèles. Il peut ainsi être noté N pour le mode nominal, D pour le mode dégradé, A pour le mode d'arrêt, I pour le mode initial, etc. ♦

⌜ *Exemple*

Dans le cas de notre exemple, le système possède, comme décrit dans le cahier des charges, deux modes (nominal et dégradé). Nous définissons alors l'ensemble de modes $\mathcal{M} = \{N, D\}$. *N est le nom représentant le mode nominal et D est le nom du mode dégradé.* ⌟

Un mode est une configuration physique particulière du système où celui-ci doit respecter un ensemble de spécifications. Dans la définition 25, nous définissons la configuration particulière qui est caractérisée par un ensemble de composants utilisés dans le mode considéré.

Définition 25 (Ensemble $\mathcal{C}^{M_j}$ des composants d'un mode M_j)
Nous définissons $\mathcal{C}^{M_j}$ comme l'ensemble des composants utilisés dans le mode M_j, où $\mathcal{C}^{M_j} = \mathcal{C}_{\circlearrowleft}^{M_j} \cup \mathcal{C}_{\leftarrow}^{M_j} \cup \mathcal{C}_{\rightarrow}^{M_j}$ tel que :

- *$\mathcal{C}_{\circlearrowleft}^{M_j}$ est l'ensemble des composants représentant le comportement* intramodal *du procédé dans le mode M_j ;*
- *$\mathcal{C}_{\leftarrow}^{M_j}$ est l'ensemble des composants générant un événement de commutation impliquant une* activation *du mode M_j ;*
- *$\mathcal{C}_{\rightarrow}^{M_j}$ est l'ensemble des composants générant un événement de commutation impliquant une* désactivation *du mode M_j ;*
- *$\mathcal{C}_{\rightleftarrows}^{M_j} = \mathcal{C}_{\leftarrow}^{M_j} \cup \mathcal{C}_{\rightarrow}^{M_j}$ est l'ensemble des composants pouvant générer un événement de commutation.*

De plus, l'ensemble de tous les composants contenus dans les modes doit être l'ensemble des composants : $\mathcal{C} = \bigcup_{M_j \in \mathcal{M}} \mathcal{C}^{M_j}$ ♦

Remarque 3
Par activation *et* désactivation*, nous sous-entendons que le mode représentant le comportement du système est celui qui est activé. Pour éviter des problèmes de cohérence, soulevés dans le chapitre 2, un seul mode est actif à la fois. Ainsi, pour n'avoir qu'un seul mode actif au fil du temps de production, les modes du système subiront une succession d'activations et de désactivations.* ♦

Il faut noter qu'il n'existe aucune relation particulière entre les ensembles de composants $\mathcal{C}_{\circlearrowleft}^{M_j}$, $\mathcal{C}_{\leftarrow}^{M_j}$ et $\mathcal{C}_{\rightarrow}^{M_j}$ excepté que tous ces ensembles sont inclus dans l'ensemble $\mathcal{C}$. En particulier, un composant peut être inclus dans :

- $\mathcal{C}_{\circlearrowleft}^{M_j}$, mais ne pas être un composant générant un événement de commutation comme ceux inclus dans $\mathcal{C}_{\rightleftarrows}^{M_j}$,
- $\mathcal{C}_{\circlearrowleft}^{M_j}$ et dans $\mathcal{C}_{\leftarrow}^{M_j}$. Cela signifie que ce composant est utilisé dans le mode M_j et génère également un événement qui active ce mode. Par contre ce composant ne génère pas d'événement qui désactive M_j.
- $\mathcal{C}_{\rightleftarrows}^{M_j}$ mais pas dans $\mathcal{C}_{\circlearrowleft}^{M_j}$. Cela signifie que ce composant n'est nécessaire pour le mode M_j que pour générer l'événement qui implique une activation ou une désactivation de ce mode.

⌜ *Exemple*

Appliquées à notre exemple, ces définitions nous donnent la figure 3.4 qui illustre une dé-

composition modale. Sur cette figure, chaque composant est représenté dans le mode dans lequel il est utilisé.

$\mathcal{M} = \{N, D\}$
$\mathcal{C} = \{C_1, C_2, C_3, C_4\}$
$\mathcal{C}^N = \{C_1, C_2, C_4\} = \mathcal{C}^N_{\circlearrowleft}$
$\mathcal{C}^N_{\rightleftarrows} = \{C_1\}$
$\mathcal{C}^D = \{C_1, C_2, C_3\}$
$\mathcal{C}^D_{\circlearrowleft} = \{C_2, C_3\}$ et $\mathcal{C}^D_{\rightleftarrows} = \{C_1\}$

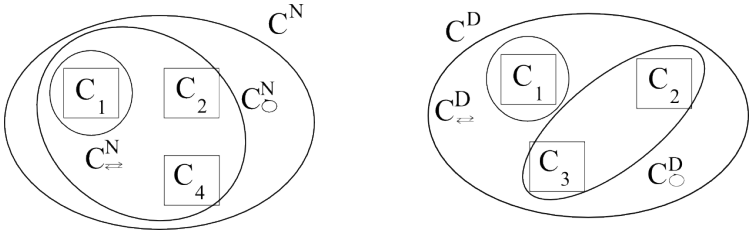

FIGURE 3.4 – Exemple d'une décomposition modale

Nous pouvons voir sur cette figure que le composant C_1 est utilisé dans le comportement interne du mode nominal, mais ne l'est pas dans celui du mode dégradé. Or il fait parti de l'ensemble des composants du mode dégradé $\mathcal{C}^D$ car le composant C_1 génère les événements f_1 et r_1 qui impliquent une activation ou une désactivation des modes nominal et dégradé. ⌟

Pour les besoins de la démarche, le concepteur doit également concevoir, à partir d'informations issues du cahier des charges, le modèle d'un automate de modes représentant le comportement commutatif entre les modes du système.

Définition 26 (Automate de modes)
Le modèle de l'automate de modes se nomme $G^{\mathcal{M}}$. Formellement, cet automate est défini par $G^{\mathcal{M}} = (Q^{\mathcal{M}}, \Sigma^{\mathcal{M}}, \delta^{\mathcal{M}}, q_0^{\mathcal{M}}, Q_m^{\mathcal{M}})$ tel que :

- $Q^{\mathcal{M}} = \mathcal{M}$
- $\Sigma^{\mathcal{M}}$ *est l'ensemble des événements de commutation du système étudié,*
- $\delta^{\mathcal{M}} : \mathcal{M} \times \Sigma^{\mathcal{M}} \to \mathcal{M}$ *est la fonction de transition de l'automate de modes,*
- $q_0^{\mathcal{M}} \in \mathcal{M}$,
- $Q_m^{\mathcal{M}} \subseteq \mathcal{M}$. ♦

Cet automate de modes est la représentation mathématique de ce qui est exprimé dans le cahier des charges. Il illustre le comportement commutatif du système. Exprimé ainsi, il est plus facile pour le concepteur d'interpréter les recommandations du cahier des charges et d'en détecter les erreurs possibles. L'automate de modes permet également de garder l'information indiquant dans quel mode le système se trouve.

⌜ *Exemple*

Le comportement commutatif du système à quatre machines et un stock est décrit section.3.2.2. Le système est représenté selon deux modes – nominal et dégradé – qui sont activés/désactivés par les événements f_1 et r_1 générés par la machine C_1. Le modèle de l'automate

de modes de notre système, en respectant la définition 26, est illustré figure 3.5. Le mode initial du système est le mode (nominal) N.

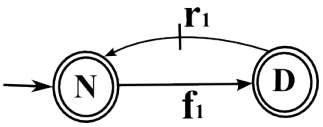

FIGURE 3.5 – Automate de mode $G^{\mathcal{M}}$

⌟

À ce stade, la plupart des modèles ont été définis et construits à partir du cahier des charges. Cependant, avant de passer à la phase de construction des modèles du système dans un mode, il convient de valider ces modèles initiaux. En effet, si une erreur se trouve dans ces premiers modèles, cette erreur se répercutera dans les prochains modèles construits et il sera très difficile de la détecter. C'est la raison pour laquelle nous procédons à une phase de validation.

3.4.2 Validation de la cohérence entre modèles

Dans l'étape de construction précédente, l'ensemble $\mathcal{M}$ est intégré à la définition de $G^{\mathcal{M}}$. Ces deux modèles sont donc obligatoirement cohérents l'un par rapport à l'autre. Par contre, les relations entre les composants G^{C_i} et les ensembles $\mathcal{C}_{\circlearrowleft}^{M_j}$, $\mathcal{C}_{\rightarrow}^{M_j}$, $\mathcal{C}_{\leftarrow}^{M_j}$ et $G^{\mathcal{M}}$ ne sont pas formellement définies. En effet, ces relations correspondent à des choix de modélisation, et donc, par définition, peuvent être multiples. Il est cependant possible de vérifier que les choix faits dans un modèle ne sont pas en contradiction avec ceux effectués dans un autre.

La définition suivante énumère les propriétés à respecter pour valider cette cohérence :

Définition 27 (Cohérence entre modèles)

Condition 1

Quel que soit l'état M_j de $\mathcal{M}$, le mode associé est accessible à partir d'un autre mode ou permet d'accéder à un autre mode, et ce mode a un comportement interne, c'est-à-dire l'ensemble $\mathcal{C}_{\circlearrowleft}^{M_j}$ n'est pas vide : $\forall M_j \in \mathcal{M},\ \mathcal{C}_{\circlearrowleft}^{M_j} \neq \emptyset \wedge \mathcal{C}_{\rightleftarrows}^{M_j} \neq \emptyset$ ♦

Condition 2

Les ensembles $\mathcal{C}_{\rightarrow}^{M_j}$ sont cohérents si : C_i est inclus dans $\mathcal{C}_{\rightarrow}^{M_j}$ ssi un événement $\alpha \in \Sigma_{\rightleftarrows}^{C_i}$ existe et qu'une transition avec α existe dans $\delta^{\mathcal{M}}$ à partir de l'état M_j :

$\forall\, M_j \in \mathcal{M},\ [C_i \in \mathcal{C}_{\rightarrow}^{M_j} \iff (\exists\, \alpha \in \Sigma_{\rightleftarrows}^{C_i},\ \exists\, M_k \in \mathcal{M},\ \delta^{\mathcal{M}}(M_j, \alpha) = M_k)]$ ♦

Condition 3

Les ensembles $\mathcal{C}_{\leftarrow}^{M_j}$ sont cohérents si : C_i est inclus dans $\mathcal{C}_{\leftarrow}^{M_j}$ ssi un événement $\alpha \in \Sigma_{\rightleftarrows}^{C_i}$ existe et qu'une transition avec α existe dans $\delta^{\mathcal{M}}$ vers l'état M_j :

$$\forall\, M_j \in \mathcal{M},\ [C_i \in \mathcal{C}_{\leftarrow}^{M_j} \iff (\exists\, \alpha \in \Sigma_{\rightleftarrows}^{C_i},\ \exists\, M_k \in \mathcal{M},\ \delta^{\mathcal{M}}(M_k, \alpha) = M_j)] \quad \blacklozenge$$

Condition 4

Les composants C_i dont $\Sigma_{\rightleftarrows}^{C_i}$ n'est pas vide doivent appartenir à au moins un ensemble $\mathcal{C}_{\rightleftarrows}^{M_j}$:

$$\forall\, C_i \in \mathcal{C}, [\Sigma_{\rightleftarrows}^{C_i} \neq \emptyset \iff \exists\, M_j \in \mathcal{M},\ C_i \in \mathcal{C}_{\rightleftarrows}^{M_j}] \quad \blacklozenge$$

Condition 5

Les événements des alphabets $\Sigma_{\rightleftarrows}^{C_i}$ des composants C_i doivent être utilisés dans l'alphabet de $G^{\mathcal{M}}$: $\Sigma^{\mathcal{M}} = \bigcup_{C_i \in \mathcal{C}} \Sigma_{\rightleftarrows}^{C_i}$ $\blacklozenge$

Dans le cas où toutes ces conditions sont respectées, la construction des modèles a été bien menée et la propriété de cohérence est validée. Cela signifie que chaque mode contient un comportement interne et un comportement commutatif (condition 1), que, pour chacun d'eux, tous les événements de commutations aient bien été pris en compte et qu'il y ait au moins un composant générant un événement permettant d'activer ce mode et un composant générant un événement permettant de le désactiver (conditions 2 à 4). Enfin, cela signifie que tous les événements de commutation appartiennent à l'ensemble des événements de l'automate de modes $G^{\mathcal{M}}$ (condition 5).

Dans le cas contraire, le concepteur peut identifier le problème suivant les conditions qui n'ont pas été respectées et, avec l'aide du client, modifier tout ou partie des modèles précédemment construits. Ainsi, si la condition 1 n'est pas respectée, alors un mode est mal défini dans le cahier des charges. Si une des conditions 2 à 4 n'est pas respectée, cela signifie que l'erreur porte sur un événement de commutation qui n'a pas été utilisé. Enfin, si c'est la condition 5 qui n'est pas respectée, alors l'automate de modes $G^{\mathcal{M}}$ a été incorrectement construit.

La validation de la cohérence des modèles est donc une première étape dans l'assurance qu'aucune erreur descriptive des modèles n'ait été faite.

⌜ *Exemple*

Nous allons vérifier la cohérence des modèles construits sur la base du système à quatre machines et d'un stock (pour rappel, l'ensemble des modes est $\mathcal{M} = \{N, D\}$ et l'automate de modes $G^{\mathcal{M}}$ est illustré figure 3.5) :

- *Explicités dans l'exemple précédent, les ensembles $\mathcal{C}_{\circlearrowleft}^{N}$, $\mathcal{C}_{\rightleftarrows}^{D}$, $\mathcal{C}_{\circlearrowleft}^{D}$ et $\mathcal{C}_{\rightleftarrows}^{D}$ ne sont pas vides. Ainsi, la condition 1 est vérifiée.*
- *Les conditions 2 et 3 permettent de s'assurer que le composant C_1 peut effectivement appartenir aux ensembles de commutation $\mathcal{C}_{\rightleftarrows}^{N}$ et $\mathcal{C}_{\rightleftarrows}^{D}$. Ceci est bien le cas et est vérifié par les fonctions de transition de l'automate de modes $\delta^{\mathcal{M}}(N, f_1) = D$ et $\delta^{\mathcal{M}}(D, r_1) = N$.*

– *Les conditions* 4 *et* 5 *assurent que les événements* f_1 *et* r_1 *sont bien présents dans l'automate de modes - ce qui est bien le cas - et que le composant* C_1 *appartienne bien à au moins un ensemble de composants de mode - ce qui est encore une fois vrai.*

Ainsi, nous sommes en mesure de dire que les modèles sont cohérents. Il est maintenant possible de passer à la phase de construction du comportement interne des modes. ⌟

3.5 Étude intramodale

L'étude intramodale, illustrée figure 3.6, est similaire à l'approche centralisée présentée dans la Théorie de Contrôle par Supervision. Son objectif est de permettre une première étude de chaque mode, indépendamment des autres. Ceci permet de vérifier que les spécifications à respecter sont bien construites et n'autorisent que le comportement interne désiré. Cette étude se fait donc sur un comportement réduit du comportement global du système, permettant une interprétation plus aisée du modèle par le concepteur et facilitant ainsi les corrections éventuelles.

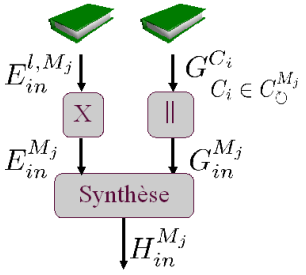

FIGURE 3.6 – Schéma de construction des modèles dans l'étude intramodale

3.5.1 Construction des procédés

Pour chaque mode M_j, nous construisons le procédé $G_{in}^{M_j}$, le modèle représentant le comportement interne dans le mode M_j, qui résulte de la composition parallèle (cf. définition 11) des automates G^{C_i} qui sont les modèles de composants utilisés dans ce mode. Formellement, nous définissons donc le modèle du procédé $G_{in}^{M_j}$ dans le mode M_j tel que $G_{in}^{M_j} = ||_{C_i \in C_{\circlearrowleft}^{M_j}} G^{C_i}$

⌜ *Exemple*

Dans le cahier des charges de l'exemple, nous avons deux modes (nominal et dégradé), ayant respectivement comme ensemble de composants $C_{\circlearrowleft}^{N} = \{C_1, C_2, C_4\}$ *et* $C_{\circlearrowleft}^{D} = \{C_2, C_3\}$.

En suivant l'opération mathématique de composition parallèle d'automates, nous avons :

– *Procédé dans le mode nominal (interne) :* $G_{in}^{N} = G^{C_1} || G^{C_2} || G^{C_4}$,

– *Procédé dans le mode dégradé (interne) :* $G_{in}^{D} = G^{C_2} || G^{C_3}$,

Les figures 3.7 et 3.8 représentent respectivement les modèles du procédé dans le mode nominal et dégradé. Ces modèles représentent le comportement possible du procédé dans chaque mode.

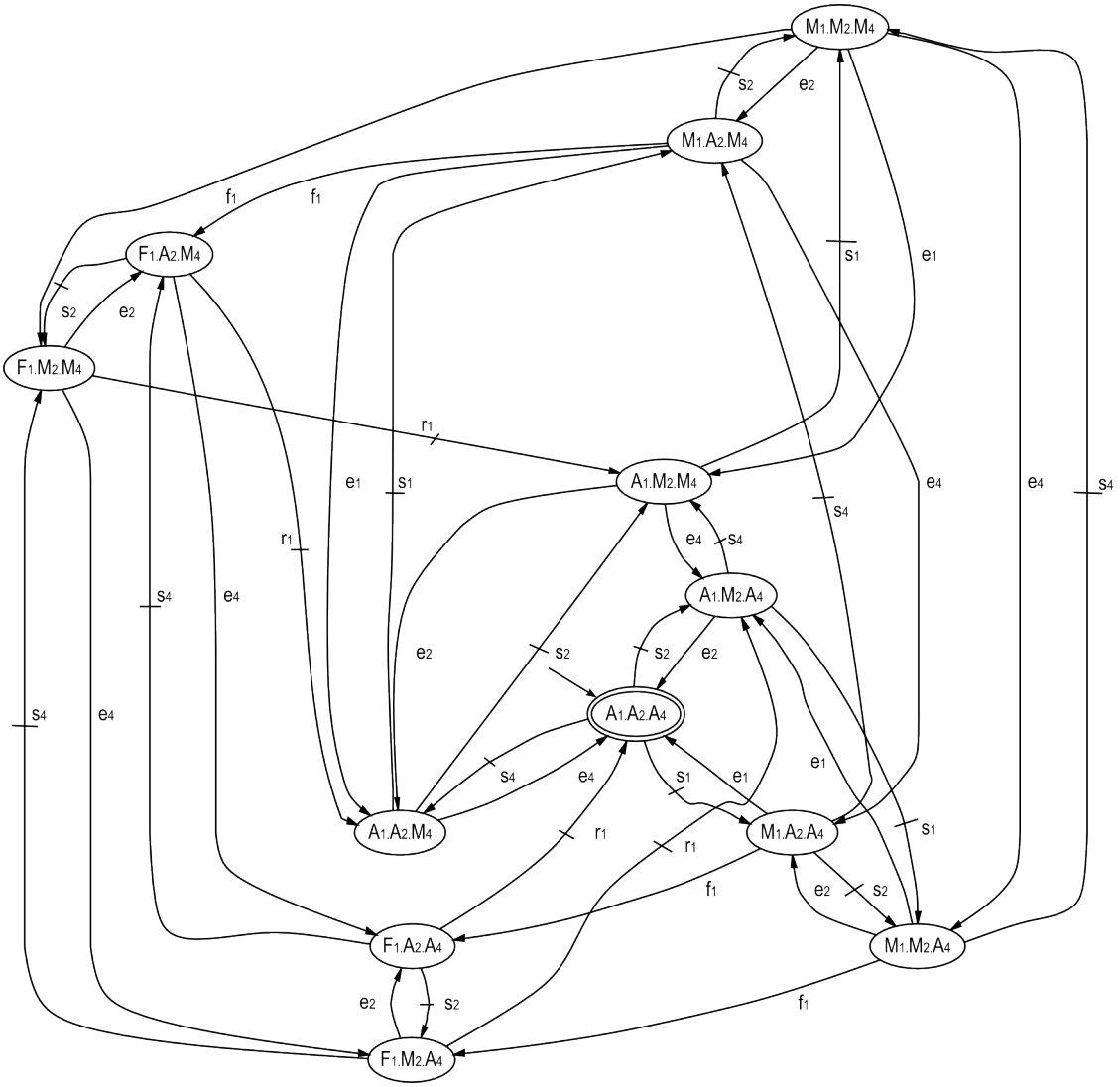

FIGURE 3.7 – Modèle du procédé interne G_{in}^{N}

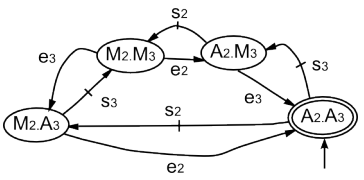

FIGURE 3.8 – Modèle du procédé interne G_{in}^{D}

3.5.2 Construction des modèles de spécifications

En plus du modèle du procédé, il est nécessaire d'avoir un modèle automate des spécifications pour effectuer une synthèse et obtenir le modèle du procédé sous contrôle. La Théorie de Contrôle par Supervision permet de construire ce modèle des spécifications par le produit (Cf. définition 12) des modèles représentant chaque spécification séparément. Ainsi, si nous appelons $E_{in}^{M_j}$ le modèle des spécifications à respecter dans le mode M_j, les modèles E_{in}^{l,M_j} (avec $l \in \mathbb{N}$ et $l \geq 1$.) sont les modèles représentant chaque spécification modélisée séparément.

⌜ *Exemple*

Dans notre exemple, la seule contrainte liée au fonctionnement interne des modes est la spécification du stock (taille maximale). Cette taille est limitée à 1 *pour notre exemple afin de limiter la taille des modèles. De plus, aucun problème en plus de ceux discutés dans ce mémoire n'apparaît lorsque la taille du stock dépasse* 1. *Ainsi, la figure 3.9 illustre les modèles de spécifications dans les modes nominal (modèle de gauche) et dégradé (modèle de droite).*

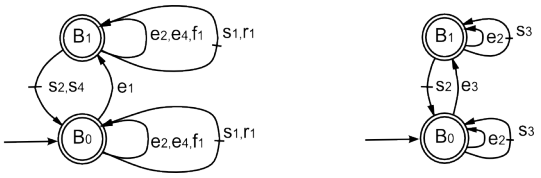

FIGURE 3.9 – Modèles de spécifications : E_{in}^{N} à gauche et E_{in}^{D} à droite

⌟

3.5.3 Procédé sous contrôle interne

À partir des modèles de procédés et de spécifications dans chaque mode, nous sommes en mesure de construire le procédé sous contrôle.

Le procédé sous contrôle $H_{in}^{M_j}$ dans le mode M_j est obtenu par la définition suivante :

Définition 28 (Procédé sous contrôle interne $H_{in}^{M_j}$)
$H_{in}^{M_j} = (Y_{in}^{M_j}, \Sigma_{in}^{M_j}, \tau_{in}^{M_j}, y_{in,0}^{M_j}, Y_{in,m}^{M_j})$ *où :* $L_m(H_{in}^{M_j}) = [L_m(G_{in}^{M_j} \times E_{in}^{M_j})]^{\uparrow c}$ ♦

Pour rappel, l'opérateur $\uparrow c$ correspond au calcul du langage suprême contrôlable qui est le plus grand sous-langage du procédé qui respecte les spécifications (cf. section 1.3.3).

⌜ *Exemple*

Dans notre exemple, nous construisons les procédés sous contrôle H_{in}^{N}, illustré figure 3.10, et H_{in}^{D}, illustré figure 3.11.

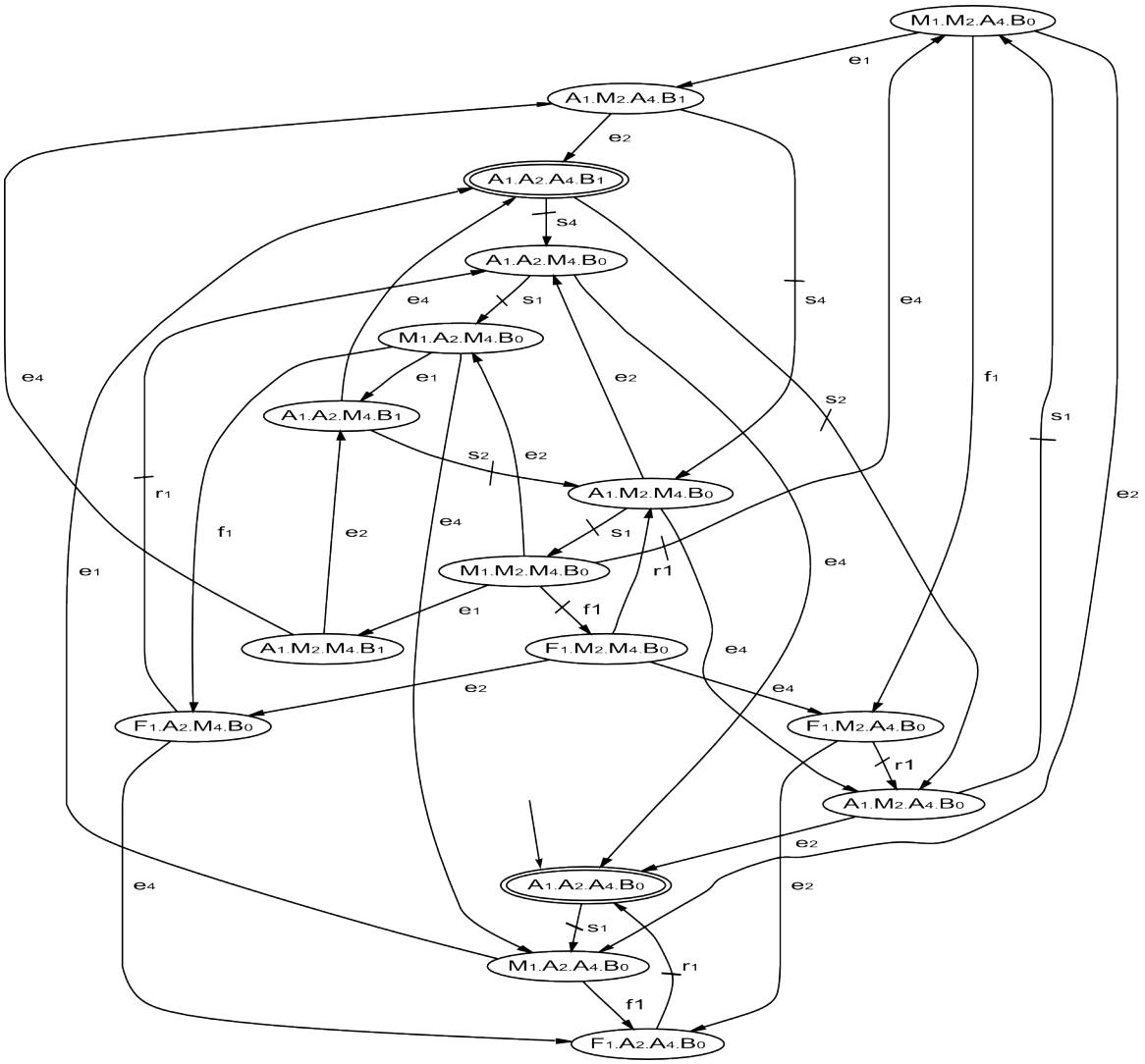

FIGURE 3.10 – Modèle du procédé sous contrôle H_{in}^{N}

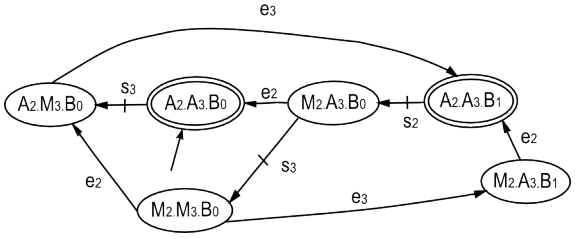

FIGURE 3.11 – Modèle du procédé sous contrôle H_{in}^{D}

3.5.4 Validation de l'étude intramodale

Durant l'étude intramodale (étape 2 de la figure 3.3, page 53), le travail principal du concepteur est la modélisation des spécifications, puis leur utilisation pour la génération automatique des procédés contrôlés. Pour être validés par le client, ces procédés contrôlés doivent exister et correspondre aux comportements attendus.

Dans le cas où le procédé sous contrôle d'un mode M_j n'existe pas, cela peut être dû à des spécifications E_{in}^{l,M_j} trop contraignantes, à une dynamique (exprimée dans G^{C_i}) trop restreinte des composants C_i, ou encore à une impossibilité de contrôler le procédé. Si le concepteur a correctement formalisé le cahier des charges, l'inexistence d'un procédé sous contrôle entraîne donc une modification de ce cahier des charges. Ceci ne peut être fait que par le client avec l'aide du concepteur. Dans le cas général, ces modifications vont porter sur les spécifications d'un mode, ce qui va entraîner une nouvelle étude intramodale, uniquement pour le mode concerné. Il peut également arriver que les modifications souhaitées portent sur la dynamique d'un composant C_i, auquel cas, c'est l'étape initiale de modélisation des exigences client qui est à refaire, ainsi que l'étude intramodale pour tous les modes utilisant ces composants. Dans le pire des cas, c'est toute l'organisation des modes qui est à revoir.

Il est également possible que les procédés sous contrôle $H_{in}^{M_j}$ existent, mais qu'ils soient considérés par le client comme beaucoup trop restreints vis-à-vis du comportement attendu, ou, inversement, comme pas assez restreints. Dans ces cas également, et avec les mêmes conséquences que précédemment, c'est le cahier des charges qui est à revoir.

Finalement, dans le cas où les procédés sous contrôles ont bien le comportement attendus et sont validés par le client, nous avons la certitude que les spécifications des modes sont respectées. Néanmoins, ces modèles ne représentent pas toutes les commutations possibles avec les autres modes. Certaines dynamiques sont manquantes, et s'intéresser à la prise en compte de toutes les commutations est l'un des objectifs de l'étape suivante.

⌜ *Exemple*

À ce stade, le concepteur et le client étudient les modèles obtenus en vue de valider ceux-ci. L'existence des modèles de procédés sous contrôle démontre qu'une solution est possible pour que le système puisse fonctionner tout en respectant les contraintes du cahier des charges. Dans notre exemple, les dynamiques obtenues correspondent à celles attendues. Nous pouvons passer à l'étape suivante qui concerne l'étude des dynamiques incluant cette fois les événements de commutation. ⌟

3.6 Étude intermodale

L'étude intramodale se concentre seulement sur les composants utilisés pour représenter le comportement interne du système dans des modes particuliers. Cela permet de s'assurer qu'un comportement respectant les spécifications internes aux modes existe.

L'objectif de l'étude intermodale, troisième étape de la démarche (figure 3.3 page 53), est d'étendre les modèles précédents afin de tenir compte de toutes les commutations possibles et de leur impact sur la dynamique des procédés, et de s'assurer que les spécifications intermodales soient respectées. Pour cela, le concepteur va premièrement lancer une phase automatique d'extension des modèles des procédés non contrôlés $G_{in}^{M_j}$ avec les modèles des composants non inclus dans $\mathcal{C}_{\circlearrowleft}^{M_j}$ qui génèrent des commutations dans ce mode. Ensuite, le concepteur doit modéliser les spécifications de commutation, précédemment non considérées, afin de générer les procédés sous contrôle H^{M_j}.

3.6.1 Extension des modèles

La construction du procédé $G_{in}^{M_j}$ de l'étude intramodale n'utilisait que les composants appartenant à $\mathcal{C}_{\circlearrowleft}^{M_j}$, c'est-à-dire les composants qui représentent le comportement interne du système dans le mode M_j. Néanmoins, ce comportement ne représente pas le comportement complet du mode car il ne tient pas compte des composants inclus dans $\mathcal{C}_{\rightleftarrows}^{M_j}$ et non inclus dans $\mathcal{C}_{\circlearrowleft}^{M_j}$. Ce sont les composants qui génèrent un événement de commutation. Ainsi, sans ces composants, toutes les trajectoires possibles de commutation ne sont pas représentées.

De plus, c'est à cette étape qu'est utilisé l'automate de modes $G^{\mathcal{M}}$ illustré figure 3.5 (page 57). En effet, comme vu dans la synthèse du chapitre 2, il est important d'inclure dans la démarche une description du comportement de la partie commande afin de limiter l'explosion combinatoire et inclure le comportement modal attendu par l'expert en générant les trajectoires de changement de modes. Le modéliser par un automate et l'inclure dans notre procédé permet d'ajouter dans le nom des états le nom du mode dans lequel le système se trouve. Avoir cette information présente de manière claire dans les modèles facilitera les futurs étapes de la démarche proposée.

Formellement, le procédé G^{M_j} est donc construit tel que :

Définition 29 *(G^{M_j})*
$G^{M_j} = (Q^{M_j}, \Sigma^{M_j}, \delta^{M_j}, q_0^{M_j}, Q_m^{M_j})$ *où :* $G^{M_j} = [||_{C_k \in \mathcal{C}^{M_j}} G^{C_k}]||G^{\mathcal{M}}$ ♦

Ces modèles assurent que le comportement *complet* du système est représenté et que *toutes* les trajectoires de commutation existent.

⌜ *Exemple*

Les ensembles de composants dans chaque mode sont :

– $\mathcal{C}^N = \{C_1, C_2, C_4\}$,

– $\mathcal{C}^D = \{C_1, C_2, C_3\}$.

En suivant l'opération mathématique de composition parallèle d'automates, nous avons :

– *Procédé dans le mode nominal :* $G^N = G^{\mathcal{M}} || G^{C_1} || G^{C_2} || G^{C_4}$,

– *Procédé dans le mode dégradé :* $G^D = G^{\mathcal{M}} || G^{C_1} || G^{C_2} || G^{C_3}$,

Nous pouvons remarquer, d'après la formule de construction, que le modèle G^N est égal au modèle G^N_{in} construit dans la section 3.5.1 [1]*. Ceci est juste et vient du fait que leurs ensembles de composants sont identiques. En effet, dans le cas du mode nominal, nous avons $\mathcal{C}^N = \{C_1, C_2, C_4\} = \mathcal{C}^N_{\circlearrowleft}$. Ainsi, dans le cas précis où $\mathcal{C}^{M_j} = \mathcal{C}^{M_j}_{\circlearrowleft}$, la phase d'extension est inutile car toutes les trajectoires sont déjà prises en compte, même celles liées aux commutations.*

Les procédés G^D, représentant le système dans le mode dégradé D, et G^N, représentant le système dans le mode nominal N, sont respectivement illustrés sur les figures 3.12 et 3.13.

Sur la figure 3.12, les lignes et états en pointillés sont ceux que la phase d'extension a ajoutés.

⌟

3.6.2 Spécifications intermodales

Les spécifications intramodales, représentées par les modèles $E^{M_j}_{in}$, sont construites uniquement sur les ensembles d'événements générés par les composants utilisés dans les modes. Du fait de l'extension des modèles de procédé avec les composants pouvant générer des événements de commutation, l'ensemble des événements des modes a changé. Il faut donc prendre en compte cet ajout d'événements dans les modèles intramodales de spécifications. On parle alors d'extension des spécifications intramodales. Ces modèles étendus seront nommés $E^{M_j}_{\circlearrowleft}$. Cette extension ne peut se faire de manière automatique (en mettant par exemple simplement en boucle sur les états les nouveaux événements) car l'ajout des événements dans les modèles dépend de l'interaction des événements déjà présents avec ceux qui vont être rajoutés. Ceci ne peut donc être fait que par un expert de manière manuelle. Toutefois, il n'est pas nécessaire de refaire complètement les modèles, ce qui est un gain de temps certains pour réaliser cette extension.

Dans cette étape, nous ajoutons également des nouveaux modèles $E^{M_j}_{\rightleftarrows}$ représentant uniquement les spécifications liées aux commutations. Ce sont les contraintes qui autorisent ou interdisent certaines commutations.

En résumé, le modèle E^{M_j}, représentant les spécifications intermodales à respecter dans le mode M_j, est construit à partir de deux types de spécifications :

1. au nom des états près

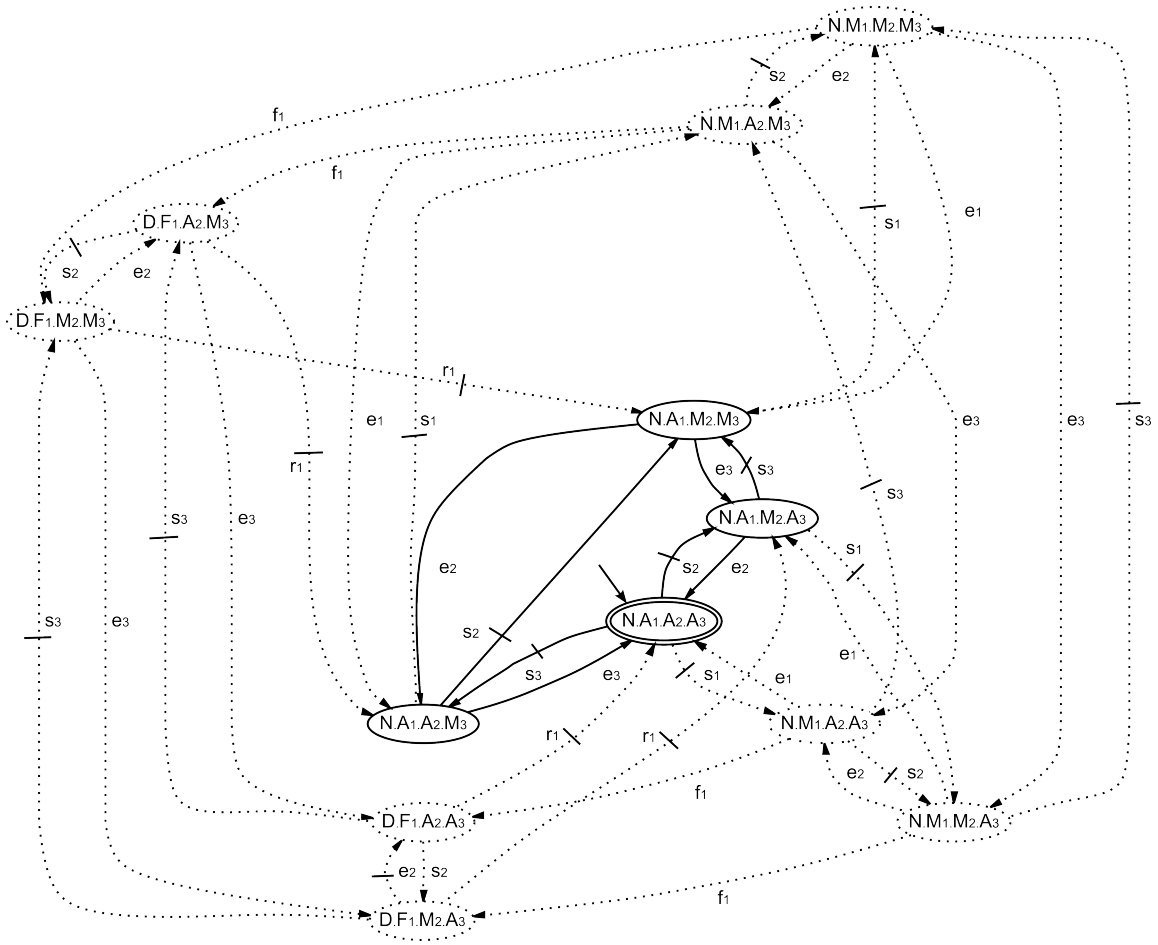

FIGURE 3.12 – Modèle du procédé G^D

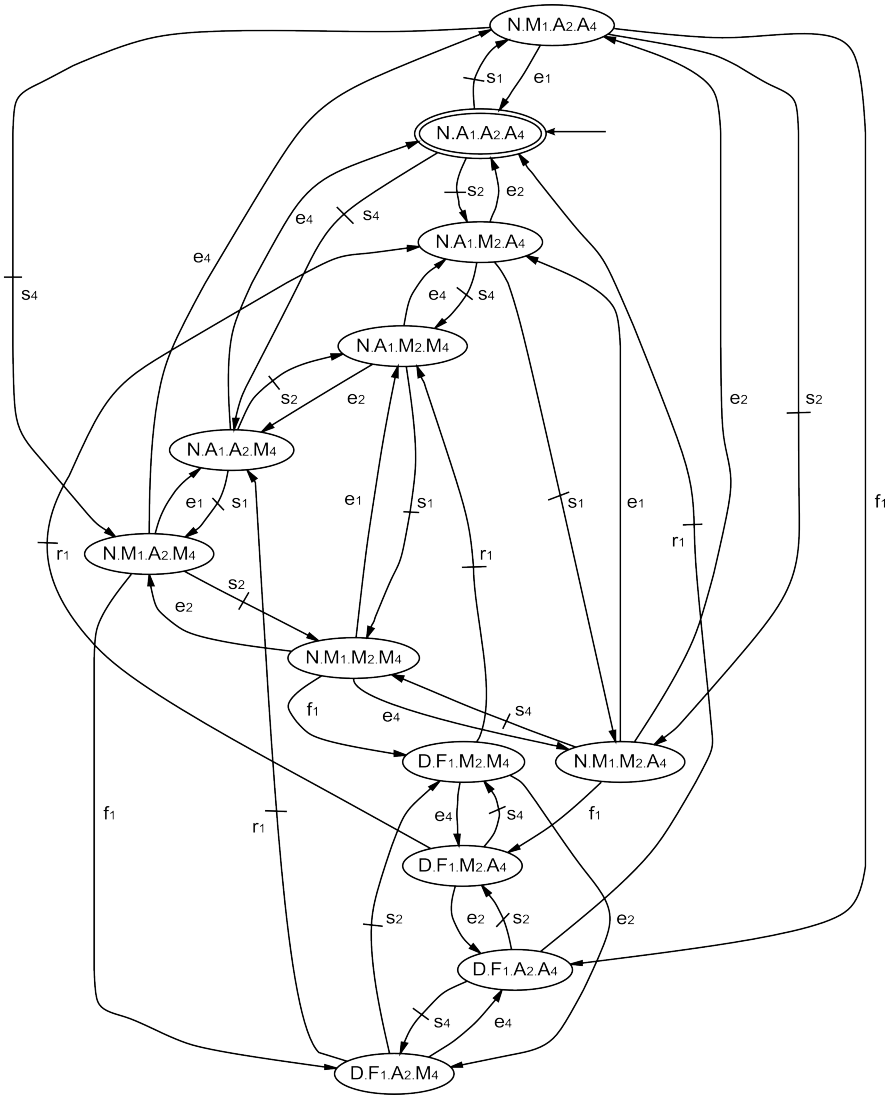

FIGURE 3.13 – Modèle du procédé G^N

1. Les spécifications intramodales étendues $E_{\circlearrowleft}^{M_j}$ eux-mêmes construites par extension des spécifications intramodales $E_{in}^{M_j}$.

2. Les spécifications de commutation représentés par $E_{\rightleftarrows}^{M_j}$ qui sont les spécifications propres à la commutation entre les modes et qui correspondent à des contraintes à respecter pour réaliser un changement de mode.

La prise en compte de l'ensemble de ces spécifications représente les spécifications intermodales. L'utilisation de ces spécifications assurent le respect des contraintes autant internes que commutatives dans chaque mode.

Formellement, la spécification E^{M_j} est construite à partir du produit des spécifications $E_{\circlearrowleft}^{M_j}$ et $E_{\rightleftarrows}^{M_j}$, eux-même construits à partir des spécifications élémentaires $E_{\circlearrowleft}^{l,M_j}$ et $E_{\rightleftarrows}^{l,M_j}$:

Définition 30 *(E^{M_j})*
E^{M_j} est défini par : $E^{M_j} = (X^{M_j}, \Sigma^{M_j}, \xi^{M_j}, x_0^{M_j}, X_m^{M_j})$ *où :* $E^{M_j} = \times_l E^{l,M_j} = (\times_l E_{\circlearrowleft}^{l,M_j}) \times (\times_l E_{\rightleftarrows}^{l,M_j})$ ♦

⌜ *Exemple*

La figure 3.14 illustre les spécifications du mode dégradé D. Le modèle $E_{\circlearrowleft}^{D}$, illustré figure 3.14(a), représente les spécifications intramodales étendues. Les transitions en pointillés sont les transitions rajoutées par le concepteur. Nous pouvons voir que la transition portant l'événement e_1 n'est pas simplement en boucle sur les états. Physiquement, les événements e_1 et e_3 remplissent le stock d'une pièce. Il ne faut donc pas négliger l'événement e_1 dans l'extension du modèle des spécifications intramodales étendues sous peine d'autoriser une surcharge de stock. Les spécifications de commutation sont représentées sur la figure 3.14(b). Cette spécification autorise ou interdit le lancement des composants C_1 ou C_3 suivant le mode dans lequel le système se trouve. Enfin, le modèle E^D résultant du produit des deux précédentes spécifications est illustré figure 3.14(c).

La figure 3.15 représente le modèle des spécifications dans le mode nominal. Dans ce mode en particulier, lié à l'exemple pris, il n'y a aucune extension à faire car tous les composants étaient déjà pris en compte, et il n'y a aucune contrainte de commutation. Le concepteur autorise toutes les commutations qui pourraient se produire (ce qui est pertinent vu que les événements de commutation sont des pannes, donc naturellement incontrôlables). Par conséquent, le modèle des spécifications intermodales E^N est identique au modèle précédemment construit des spécifications intramodale E_{in}^N.

⌟

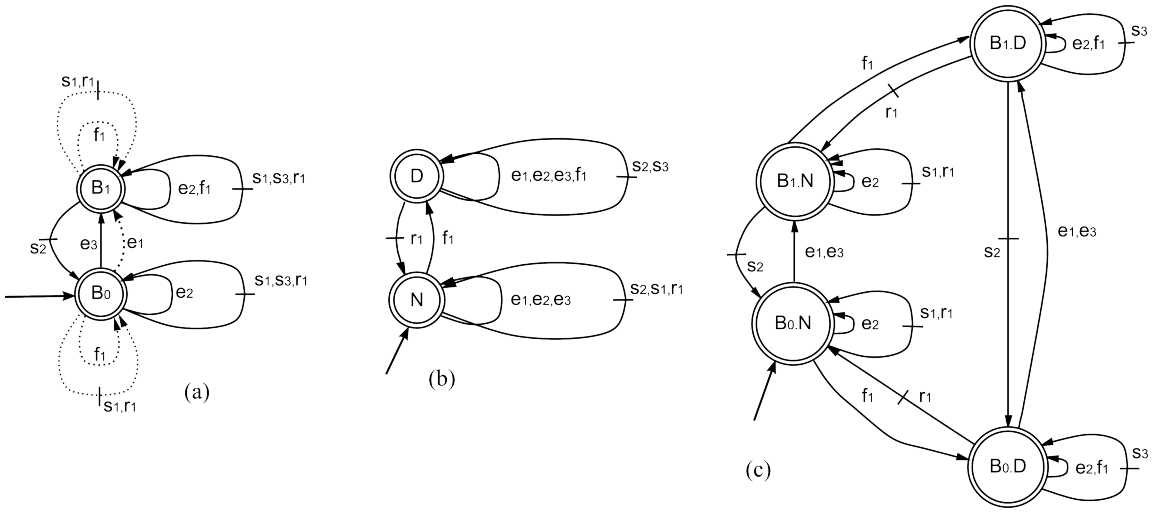

FIGURE 3.14 – (a) Spécification intramodale étendue $E^D_{\circlearrowleft}$; (b) Spécification de commutation $E^D_{\leftrightarrows}$; (c) Spécification intermodale E^D

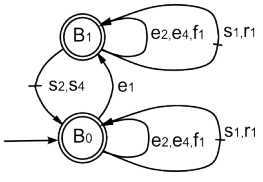

FIGURE 3.15 – Spécification E^N

3.6.3 Procédés sous contrôle étendus

Le concepteur est muni des modèles des procédés G^{M_j} dans les modes considérés et des modèles de spécifications E^{M_j}. Celui-ci peut donc construire les procédés sous contrôle H^{M_j} qui représentent les comportements admissibles des procédés étendus et qui respectent les contraintes à chaque mode.

Formellement, la synthèse de H^{M_j} s'effectue comme dans la section 1.3, et telle que :

Définition 31 *(H^{M_j})*
$H^{M_j} = (Y^{M_j}, \Sigma^{M_j}, \tau^{M_j}, y_0^{M_j}, Y_m^{M_j})$ *où :* $L_m(H^{M_j}) = [L_m(G^{M_j} \times E^{M_j})]^{\uparrow c}$ ♦

Remarque 4
Comme nous pouvons le constater, il est possible de construire des modèles qui représentent le comportement admissible du système dans chaque mode – notamment le comportement commutatif admissible au regard des contraintes. À ce stade, rien n'indique une cohérence d'un point de vue commutatif entre les modèles de mode du système. ♦

⌜ *Exemple*

La figure 3.16 illustre le modèle du procédé sous contrôle dans le mode nominal. Cependant, du fait de l'égalité des ensembles de composants ($\mathcal{C}^{M_j} = \mathcal{C}_{\circlearrowleft}^{M_j}$) et de l'inexistence de spécification de commutation ($E^N = E_{in}^N$), le modèle H^N est identique au modèle H_{in}^N. La figure 3.17 présente le modèle du procédé sous contrôle dans le mode dégradé qui, lui, a un comportement différent lié à l'ajout d'un composant (C_1) et à des spécifications de commutation (le composant C_3 ne démarre que si le composant C_1 est en panne) qui n'étaient pas pris en compte dans le comportement interne durant l'étude intramodale. Les lignes en pointillé sur cette figure sont les états et événements qui n'existaient pas dans le comportement de H_{in}^D. Nous voyons ainsi que de l'état initial, le seul événement sortant est une trajectoire en pointillé portant l'événement s_1. Ceci est conforme aux spécifications précédemment construites où le composant C_1 fonctionne avant C_3.

⌟

3.6.4 Validation de l'étude intermodale

Le concepteur a une nouvelle fois l'opportunité de valider les modèles construits. C'est la phase de validation de l'étape 3 de notre démarche. Comme dans l'étape de validation de l'étude intramodale, les deux questions qui se posent portent sur l'existence des modèles H^{M_j} et sur le comportement attendu par le client.

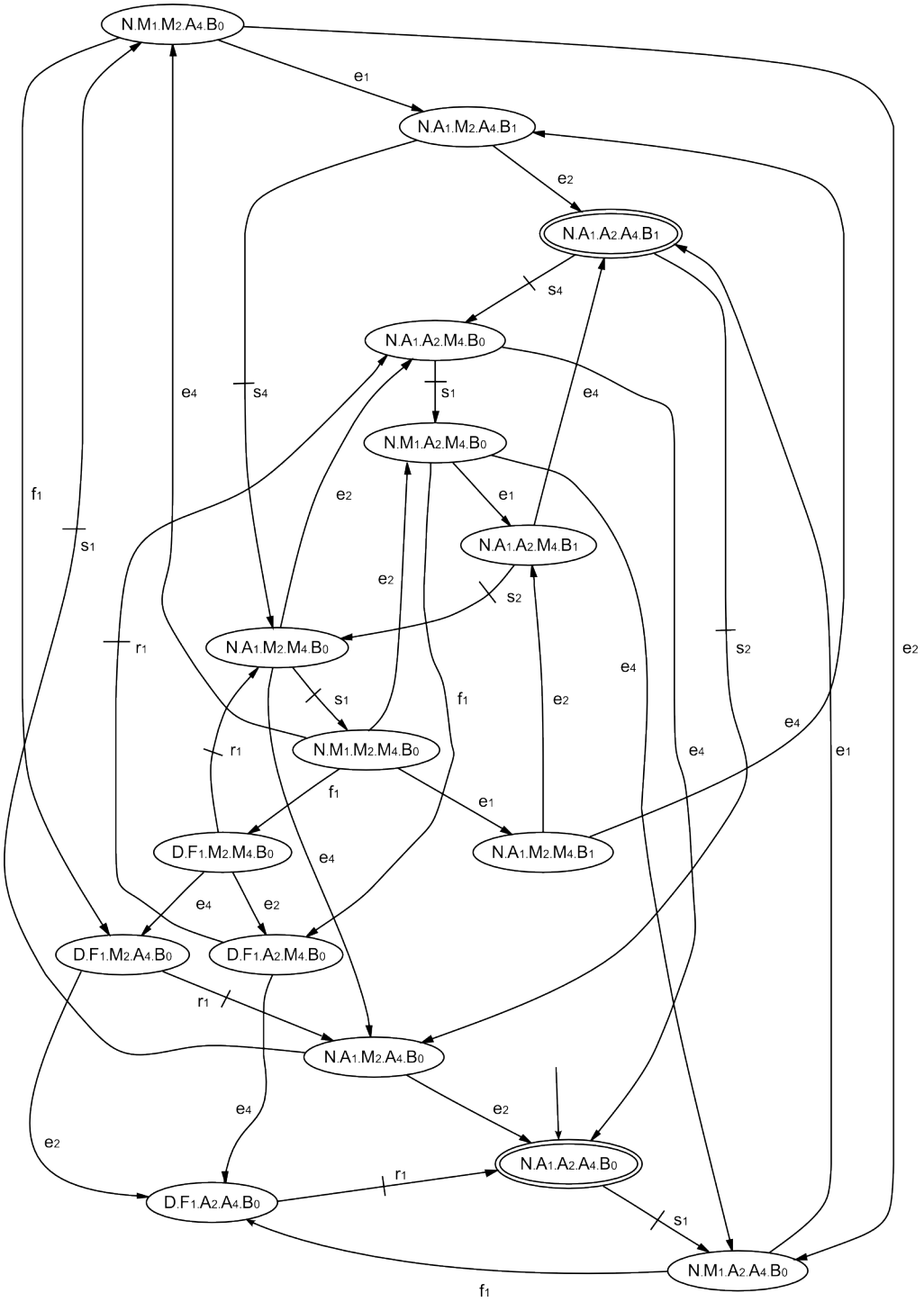

FIGURE 3.16 – Procédé sous contrôle H^N

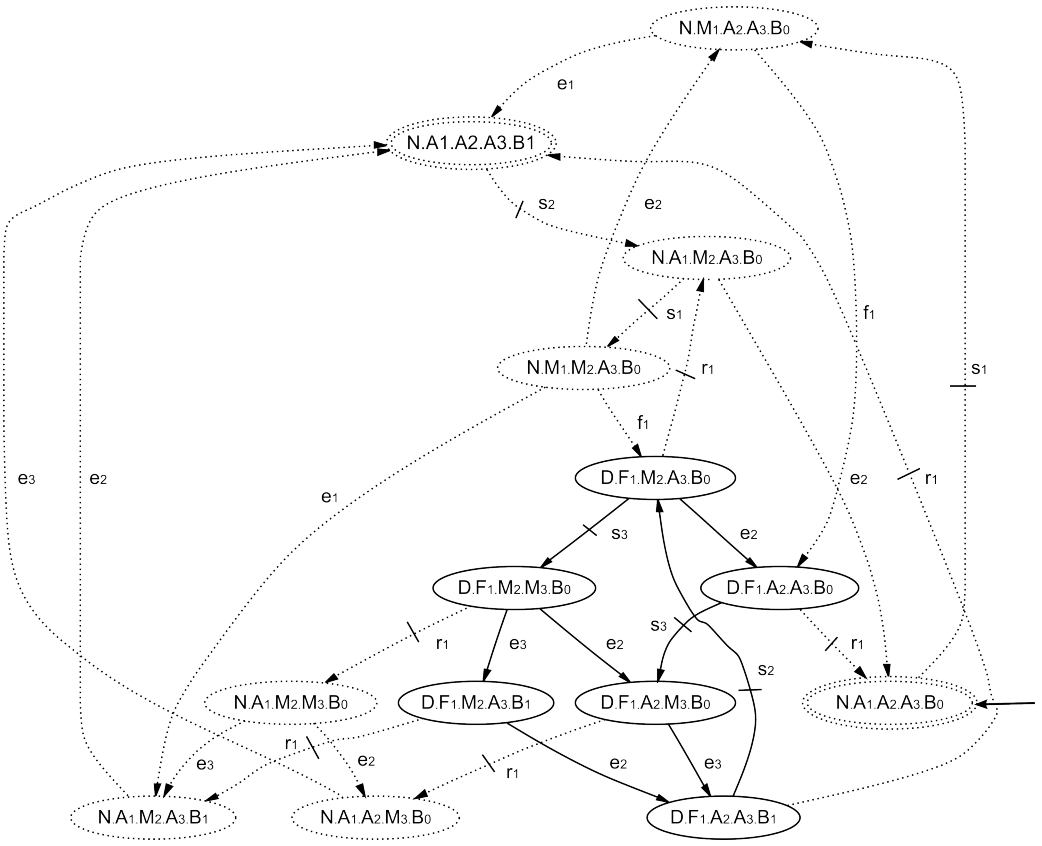

FIGURE 3.17 – Procédé sous contrôle H^D

Si les modèles H^{M_j} n'existent pas, il faudra revoir les spécifications de commutation car ce sont avant tout ces spécifications qui peuvent poser problème. En effet, les modèles G^{C_i} et $G^{\mathcal{M}}$ ont déjà été validés. Dans la pratique, ces spécifications sont très simples à écrire et il est donc peu probable que, si ces contraintes empêchent effectivement des commutations d'avoir lieu, le modèle de gestion des modes tout entier soit à revoir. Bien entendu, ceci implique alors la modification, par le client, du cahier des charges.

Dans le cas où tous les modèles H^{M_j} existent, il se peut encore une fois que le comportement du procédé sous contrôle soit pour le client beaucoup trop restreint. Cette restriction n'a pu qu'être provoquée par des spécifications de commutation. Il importe donc au concepteur d'identifier avec le client les comportements posant problème, et de proposer des spécifications empêchant les comportements non désirés afin d'obtenir un comportement du procédé sous contrôle valide. Dans l'exemple proposé, aucun comportement non désiré n'est généré, l'étude intermodale est par conséquent validée.

Lorsque tous les modèles du procédé sous-contrôle H^{M_j} ont été validés, l'étape suivante a pour objectif d'identifier toutes les occurrences d'événements de commutation et de s'assurer que toutes les commutations sortantes d'un mode ont bien des commutations entrantes équivalentes, c'est-à-dire que les commutations connectent les modes sans erreurs entre eux.

3.7 Suivi de trajectoires

Ceci est la quatrième étape, l'avant dernière, de notre démarche illustrée par la figure 3.3 page 53.

3.7.1 Définitions préliminaires

Jusqu'à présent, l'étude de chaque mode était quasi indépendante des autres modes. Cependant, lorsque le procédé contrôlé passe d'un mode M_j à un mode M_k, il faut que cette dynamique soit modélisée par une même trajectoire dans le modèle H^{M_j} et dans le modèle H^{M_k}. Il est donc nécessaire, pour chaque transition de $G^{\mathcal{M}}$, de faire un suivi de trajectoires permettant de rechercher les trajectoires de H^{M_j} conduisant à un état où est généré un événement de commutation vers H^{M_k}, de rechercher les états atteints dans le mode d'arrivée par ces trajectoires et de vérifier que pour chaque commutation il n'existe qu'un état unique. Une trajectoire qui active un mode doit toujours mettre le système dans un même état physique, ou alors il y a un problème de déterminisme sur la commutation. Enfin, avant de passer à la dernière étape de fusion d'états, il est opéré une phase d'ajout d'indice sur le nom des événements de commutation afin d'éviter l'indéterminisme provoqué par une fusion d'états. Ainsi, cette dernière phase anticipe un problème futur dans la dernière étape de notre démarche.

Formellement, la procédure de suivi de trajectoires utilise les définitions suivantes afin de

déterminer les états équivalents entre les modes. La première de ces définitions porte sur les sous-ensembles de Σ nécessaires aux autres définitions.

Définition 32 (Ensembles d'événements de commutation)
$\Sigma_{\leftrightarrows}^{M_i} \subset \Sigma^{M_i}$ *est l'ensemble des événements de commutation du mode* M_i *tel que :* $\Sigma_{\leftrightarrows}^{M_i} = \bigcup_{n \in \mathcal{C}_{\leftrightarrows}^{M_i}} \Sigma_{\leftrightarrows}^{\mathcal{C}_n}$.
$\Sigma_{\leftrightarrows}^{M_i} = \Sigma_{\leftarrow}^{M_i} \cup \Sigma_{\rightarrow}^{M_i}$ *où* $\Sigma_{\leftarrow}^{M_i}$ *(resp.* $\Sigma_{\rightarrow}^{M_i}$*) est l'ensemble des événements qui activent (resp. désactivent) le mode* M_i*. Ils sont définis tels que :*

- $\Sigma_{\rightarrow}^{M_i} = \{\alpha \in \Sigma_{\leftrightarrows}^{M_i} | M_k \in Q^{\mathcal{M}}, \delta^{\mathcal{M}}(M_i, \alpha) = M_k \text{ est défini}\}$;
- $\Sigma_{\leftarrow}^{M_i} = \{\alpha \in \Sigma_{\leftrightarrows}^{M_i} | M_k \in Q^{\mathcal{M}}, \delta^{\mathcal{M}}(M_k, \alpha) = M_i \text{ est défini}\}$; ♦

De cette définition, nous pouvons définir l'ensemble d'états $Y_{M_j \xrightarrow{\alpha} M_k}^{M_j}$ appartenant à H^{M_j} où un événement de commutation α existe et implique une commutation du mode de départ M_j vers le mode d'arrivée M_k.

Définition 33 (État de commutation)
$Y_{M_j \xrightarrow{\alpha} M_k}^{M_j} = \{y \in Y^{M_j} \mid M_j, M_k \in Q^{\mathcal{M}},\ \alpha \in \Sigma_{\rightarrow}^{M_i} \cap \Sigma_{\leftarrow}^{M_k},\ \tau^{M_j}(y, \alpha) \text{ est défini}\}$ ♦

Nous pouvons également définir le langage $L_{M_j \xrightarrow{\alpha} M_k}^{y}(H^{M_j})$, un sous-langage de $L(H^{M_j})$, qui contient tous les mots menant de l'état initial y_0 de H^{M_j} vers un état y où un événement de commutation α peut se produire.

Définition 34 (Langage de commutation)
$L_{M_j \xrightarrow{\alpha} M_k}^{y}(H^{M_j}) = \{s \in \Sigma^{M_j *} \mid y \in Y_{M_j \xrightarrow{\alpha} M_k}^{M_j},\ \tau(y_0, s) = y \text{ est défini}\}$ ♦

Ce langage correspond à toutes les dynamiques dans le mode M_j qui mènent à un état y où un des événements de commutation étudiés peut être généré. Il importe maintenant de savoir si ces dynamiques existent dans le mode d'arrivée M_k. Pour cela, nous utilisons la fonction de projection de la Théorie des Langages, étendues à deux langages distincts [Kam05b]. Cette fonction étendue, nommée P_{M_j,M_k}, est définie par :

Définition 35 (Projection étendue)
Considérons $P_{M_j,M_k} : \Sigma^{M_j *} \rightarrow \Sigma^{M_k *}$ *tel que* $\forall \sigma \in \Sigma^{M_j}$ *et* $\forall s \in \Sigma^{M_j *}$ *:*

$$P_{M_j,M_k}(\varepsilon) = \varepsilon$$
$$P_{M_j,M_k}(s\sigma) = \begin{cases} P_{M_j,M_k}(s)\sigma \text{ si } \sigma \in \Sigma^{M_j} \cap \Sigma^{M_k} \\ P_{M_j,M_k}(s) \quad \text{si } \sigma \in \Sigma^{M_j} \backslash \Sigma^{M_k} \end{cases}$$

Littéralement, la fonction de projection étendue, définie pour la première fois dans [Kam04], prend un langage défini sur l'alphabet utilisé dans le mode M_j (l'alphabet dans le mode de départ), et *efface* les événements qui ne sont pas inclus dans l'alphabet utilisé dans le mode (d'arrivée) M_k. Ceci permet d'obtenir ces dynamiques dans le mode M_k.

Cependant, avoir identifié ces dynamiques par projection ne signifie pas que ces dynamiques existent dans le mode d'arrivée. La définition suivante recherche donc si les dynamiques existent dans le langage généré par le système dans le mode M_k.

Définition 36 (Langage équivalent)
En considérant que le mode de départ soit le mode M_j et que le mode d'arrivée soit le mode M_k, $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})$ est le sous-langage de H^{M_j} menant à l'état y où un événement de commutation α peut être généré. $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k})$ est le langage projeté sur l'alphabet du mode d'arrivée M_k, tel que : $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) = P_{M_j,M_k}[L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})]$. ♦

En utilisant cette dernière définition, nous sommes en mesure de définir la propriété de compatibilité qui vérifie que si tous les langages $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k})$ sont inclus dans $L(H^{M_k})$, alors *au moins* un état de connexion existe dans $L(H^{M_k})$ pour chaque état y. Dans ce cas, H^{M_k} est compatible avec H^{M_j} (ce qui n'implique pas que H^{M_j} soit compatible avec H^{M_k}).

Définition 37 (Compatibilité entre modèles)
H^{M_k} est compatible avec H^{M_j} ssi : $\forall y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$, $(L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \subseteq L(H^{M_k}))$ ♦

La propriété de compatibilité nous permet de vérifier que les dynamiques conduisant à une commutation dans le mode de départ existent également dans le mode d'arrivée, ceci assurant une commutation effective. Cependant, de par la fonction de projection qui efface une partie de la dynamique, il est possible que plusieurs dynamiques *différentes* dans le mode de départ deviennent *identiques* dans le mode d'arrivée. Cela revient à dire que différents états du système dans le mode de départ correspondent à un même état dans le mode d'arrivée. Nous appelons cela une *inconsistance*. Elle est due à une perte d'information durant la commutation. Or, perdre une information durant une commutation signifie ne plus être en mesure de la retrouver plus tard et donc prendre le risque d'avoir des erreurs de conception. Par conséquent, il faut s'assurer qu'il existe une *consistance* dans l'état du système lors d'une commutation d'un mode à l'autre et donc vérifier qu'à une dynamique du mode d'arrivée ne corresponde qu'une seule dynamique du mode de départ. Ceci est indispensable pour être en mesure d'effectuer une commutation et de garder un état équivalent pour le système. Cette propriété est appelée *propriété de consistance* et est validée par la définition suivante :

Définition 38 (Consistance entre modèles)
Supposons que le procédé sous contrôle H^{M_k} soit compatible avec H^{M_j}. H^{M_k} est consistant avec H^{M_j} ssi :

$$\forall y_1, y_2 \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k},\ (y_1 \neq y_2 \Leftrightarrow L^{y_1}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \cap L^{y_2}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) = \emptyset) \qquad \blacklozenge$$

Cette condition signifie qu'en considérant deux langages dans M_j qui mènent à une commutation, alors leur langage projeté sur l'alphabet du mode M_k ne peuvent conduire à un même état. Cela permet de s'assurer que si le système peut être dans deux états physiques différents *avant* une commutation, alors ce système doit pouvoir être dans deux états physiques différents *après* une commutation. Sinon, un état physique du mode M_k correspondrait à deux états physiques du système après une commutation. Cela est impossible est l'un des deux est forcément une mauvaise représentation de l'état du système.

3.7.2 Procédure de suivi de trajectoires

Avec les propriétés ci-dessus, nous sommes en mesure de détailler la procédure qui identifie les couples d'états équivalents qui connectent le mode de départ M_j au mode d'arrivée M_k :

Procédure 2 (Transformation de H^{M_j} en $H^{M_j}_{lab}$)
Pour chaque $\delta^{\mathcal{M}}(M_j, \alpha) = M_k$ de l'automate de modes $G^{\mathcal{M}}$:

1. *pour chaque $y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$ [définition 33] :*
   (a) *nous calculons $L^{y}_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})$ [définition 34] ;*
   (b) *nous calculons $L^{y}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k})$ [définition 36].*
   (c) *nous vérifions la propriété de compatibilité [définition 37] :*
      i. *$(L^{y}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \not\subseteq L(H^{M_k}))$. Cela signifie que H^{M_k} n'est pas compatible avec H^{M_j}, c'est-à-dire qu'il est possible qu'une commutation survenant dans le mode de départ (et désactivant celui-ci) ne conduise pas (en l'activant) au mode d'arrivée. Il est donc inutile de continuer la procédure plus loin, la procédure est interrompue, il faut corriger ce problème ;*
      ii. *$(L^{y}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \subseteq L(H^{M_k}))$: H^{M_k} est potentiellement[2] compatible avec H^{M_j}. Nous devons vérifier la consistance [définition 38] :*
         A. *$\exists y_1, y_2 \in Y^{M_k}\ [L^{y}_{M_j \xrightarrow{\alpha} M_k}(H^{M_j}) \subseteq L^{y_1}(H^{M_k})$ et $L^{y}_{M_j \xrightarrow{\alpha} M_k}(H^{M_j}) \subseteq L^{y_2}(H^{M_k})]$ ou*
         $\exists y' \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}\ (y \neq y' \Leftrightarrow L^{y}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \cap L^{y'}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \neq \emptyset)$: H^{M_k} n'est pas consistant avec H^{M_j} et la procédure peut s'arrêter ici ;

2. Pour être compatible, il faut que la procédure ait été faite sur toutes les dynamiques de commutation. Une seule commutation incompatible implique que les modèles ne sont pas compatible.

B. sinon, H^{M_k} est potentiellement[3] consistant avec H^{M_j}. $\delta^{\mathcal{M}}(M_j,\alpha) = M_k$ est donc considérée comme valide. Nous ajoutons un indice sur l'événement de commutation utilisé dans les fonctions de transitions de H^{M_j} et de H^{M_k} tel que $\tau^{M_j}(y,\alpha)$ et $\tau^{M_k}(y',\alpha)$ existent et sont changées par $\tau^{M_j}(y,\alpha_l)$ et $\tau^{M_k}(y',\alpha_l)$ avec l l'indice. Le nouvel alphabet issue de cette étape de labélisation est défini par : $\Sigma^{M_j}_{lab} = \Sigma^{M_j} \cup \{\alpha_l\}$;

2. *Après le dernier $y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$, si la procédure s'est déroulée complètement sans avoir été interrompue pour incompatibilité ou inconsistance, alors nous pouvons dire que H^{M_k} est consistant avec H^{M_j} au regard de l'événement de commutation α. Nous pouvons étudier la commutation suivante $\delta^{\mathcal{M}}(M_j,\alpha') = M_k$* ♦

La phase d'ajout d'un indice est utile pour éviter l'indéterminisme qui est provoqué par l'étape suivante de fusion d'états. Pour rappel, nous désirons que dans chaque modèle représentant le système dans un mode, les événements de commutation activent ou désactivent de manière adéquate les modes. Pour éviter une incohérence de fonctionnement, cette désactivation est possible grâce à l'ajout d'un état puits dans lequel arrivent tous les événements de commutation qui désactivent le mode et dont partent tous les événements de commutation qui l'activent. Or, de cet état puits, un même événement de commutation peut activer notre mode dans des états différents. Ceci est dû à la méconnaissance de ce qui s'est passé lorsque le mode était inactif. Notre suivi de trajectoire permet d'identifier les événements qui font sortir le mode d'arrivée de son état puits, et de les discriminer suivant l'évolution passée. Cette discrimination se fait par l'ajout d'un indice sur les événements de commutation.

3.7.3 Consistance des modèles et validation

Durant cette procédure de suivi de trajectoire, deux phases de caractérisation (compatibilité et consistance) ont été opérées. Certaines trajectoires permettent de définir que M_k est inconsistant ou incompatible avec M_j. Dans ces deux cas, l'inconsistance ou l'incompatibilité des modèles vient d'un problème entre les spécifications des modes. Le client devra donc envisager de modifier ces spécifications, dans l'étude intermodale. Si vraiment aucune solution n'est trouvée, alors il n'est pas possible avec l'équipement actuel ou les connexions de modes définis par l'automate de modes de respecter toutes les spécifications et le client et le concepteur devront revenir sur les modèles du procédé ou de l'automate de modes.

Si la procédure de suivi de trajectoire s'est déroulée correctement, cela signifie que les modèles sont consistants et compatibles entre eux tels que définis respectivement par les définitions 38 et 37. Nous sommes donc en mesure de certifier que toutes les commutations entre les

3. Encore une fois, pour être consistant il faut que la procédure soit menée complètement

modes débouchent sur une désactivation du mode de départ et une activation du mode d'arrivée tout en respectant les spécifications requises dans les deux modes.

⌜ *Exemple*

Dans la procédure 2, nous commençons à identifier toutes les trajectoires de commutation en nous basant sur l'automate de modes $G^{\mathcal{M}}$ *illustré figure 3.5, c'est-à-dire en étudiant la fonction de transition sous la forme* $\delta^{\mathcal{M}}(M_j,\alpha)=M_k$. *Il apparaît ainsi deux transitions de commutation* $\delta^{\mathcal{M}}(N,f_1)=D$ *et* $\delta^{\mathcal{M}}(D,r_1)=N$.

Nous allons étudier toutes les commutations résultantes de la génération de l'événement f_1 *menant du mode nominal N au mode dégradé D. La même opération est faite au regard de l'événement* r_1 *menant du mode dégradé D au mode nominal N.*

Commençons avec l'étude des commutations du mode N au mode D. Pour cela, la figure 3.18 (page 79) représente le modèle H^N *du procédé sous contrôle dans le mode N. Cette figure est colorisée de manière à faire apparaître les états indiquant le mode (en* blanc *pour le mode N et en* gris *pour le D) dans lequel le système se trouve. Les trajectoires d'étude indiquant la génération de l'événement* f_1 *sont également indiquées par des pointillés.*

Les états appartenant au mode N où un événement de commutation f_1 *peut se produire et mener à un état appartement au mode D sont au nombre de* 4. *Ce sont les états :*

- N,M_1,A_2,A_4,B_0
- N,M_1,A_2,M_4,B_0
- N,M_1,M_2,A_4,B_0
- N,M_1,M_2,M_4,B_0

Pour chacun de ces états, il convient de calculer les langages $L^y_{N\overset{f_1}{\to}D}(H^N)$ *qui sont projetés sur l'ensemble des événements du mode d'arrivée D afin d'obtenir les langages* $L^y_{N\overset{f_1}{\to}D}(H^D)$. *Ces langages sont ensuite caractérisés par la propriété de compatibilité [définition 37] et de consistance [définition 38].*

Le tableau 3.1 donne, pour chaque état de $Y^N_{N\overset{f_1}{\to}D}$ *mentionné dans la première colonne, certains mots appartenant aux langages* $L^y_{N\overset{f_1}{\to}D}(H^N)$ [4] *(deuxième colonne). La troisième colonne correspond aux langages projetés sur l'alphabet du mode D.*

Nous voyons dans ce tableau, notamment dans la dernière colonne, que tous les événements générés par le composant C_4 *ont disparu lors de la projection. Ceci est en accord avec nos définitions. En effet, le composant* C_4 *n'étant pas utilisé dans le mode (dégradé) D, il est inutile d'en tenir compte pour la commutation. La tableau 3.2 reprend la première et troisième colonne du tableau 3.1, mais indique en colonne* 3 *et* 4 *si le mot est compatible et précise, si tel est le cas, l'état que ce mot atteint.*

Le tableau 3.2 met en évidence deux états compatibles (et consistants) et deux états incompatibles. Il s'agit des états (N,M_1,A_2,M_4,B_0) *et* (N,M_1,M_2,M_4,B_0). *En regardant de plus*

4. Ces langages pouvant être infinis, nous n'exprimons que certains d'entre eux permettant d'illustrer le suivi de trajectoire

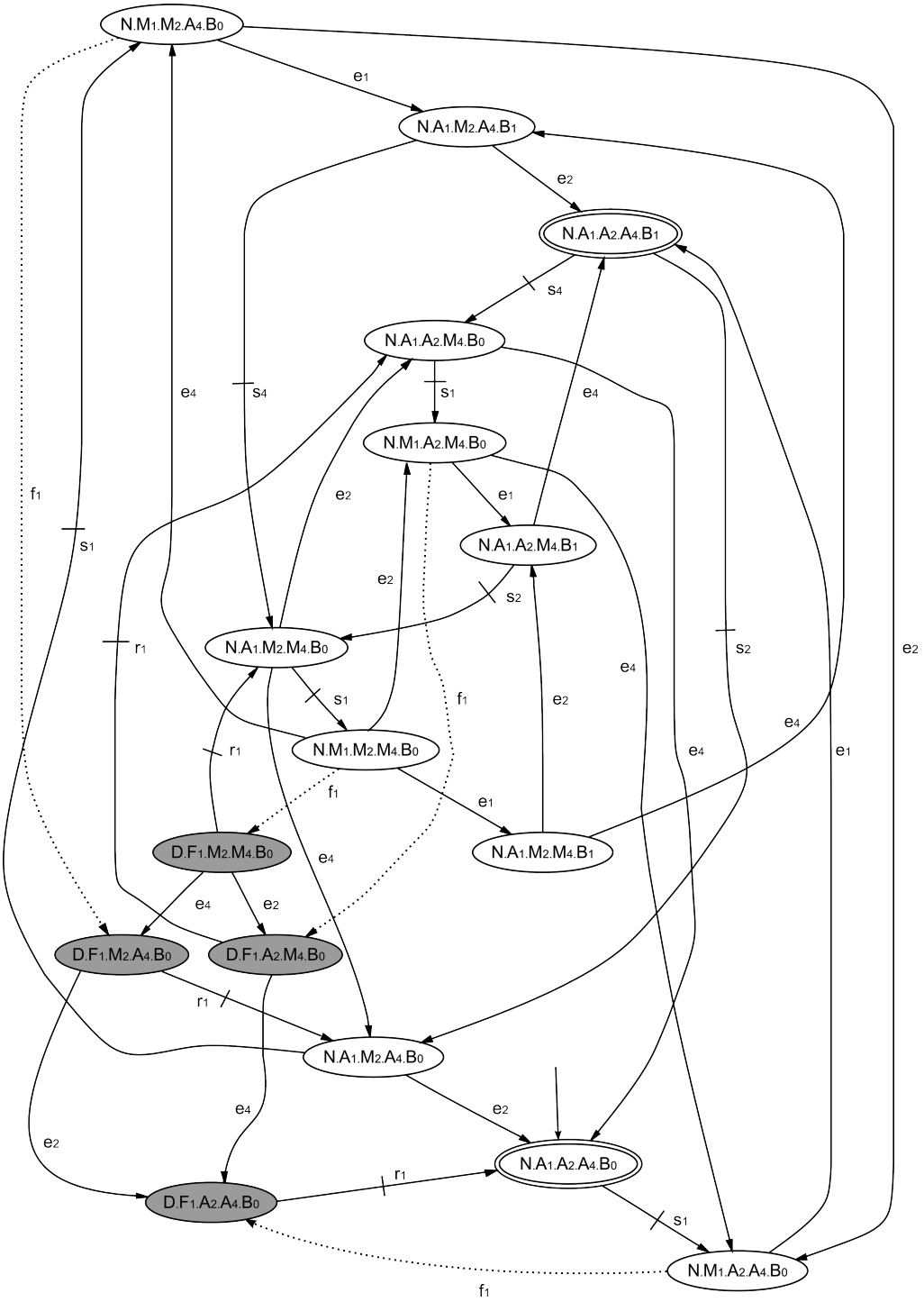

FIGURE 3.18 – Modèle du procédé sous contrôle H^N.

$y \in Y^N_{N \xrightarrow{f_1} D}$	$s \in L^y_{N \xrightarrow{f_1} D}(H^N)$	s' tel que $s' = P_{N,D}(s)$
N, M_1, A_2, A_4, B_0	s_1	s_1
	$s_1 \cdot e_1 \cdot s_2 \cdot e_2 \cdot s_1$	$s_1 \cdot e_1 \cdot s_2 \cdot e_2 \cdot s_1$
	$s_1 \cdot e_1 \cdot s_4 \cdot s_1 \cdot e_4$	$s_1 \cdot e_1 \cdot s_1$
N, M_1, A_2, M_4, B_0	$s_1 \cdot e_1 \cdot s_4 \cdot s_1$	$s_1 \cdot e_1 \cdot s_1$
	$s_1 \cdot e_1 \cdot s_4 \cdot s_1 \cdot e_1 \cdot e_4 \cdot s_4 \cdot s_1$	$s_1 \cdot e_1 \cdot s_1 \cdot e_1 \cdot s_1$
N, M_1, M_2, A_4, B_0	$s_1 \cdot e_1 \cdot s_2 \cdot s_1$	$s_1 \cdot e_1 \cdot s_2 \cdot s_1$
	$s_1 \cdot e_1 \cdot s_4 \cdot s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot e_4$	$s_1 \cdot e_1 \cdot s_1 \cdot s_2 \cdot s_1$
	$s_1 \cdot e_1 \cdot s_4 \cdot s_1 \cdot e_1 \cdot s_2 \cdot e_4 \cdot s_1$	$s_1 \cdot e_1 \cdot s_1 \cdot e_1 \cdot s_2 \cdot s_1$
N, M_1, M_2, M_4, B_0	$s_1 \cdot e_1 \cdot s_4 \cdot s_1 \cdot e_1 \cdot s_2 \cdot s_1$	$s_1 \cdot e_1 \cdot s_1 \cdot e_1 \cdot s_2 \cdot s_1$
	$s_1 \cdot e_1 \cdot s_4 \cdot s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot e_1 \cdot e_2 \cdot s_2 \cdot s_1$	$s_1 \cdot e_1 \cdot s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot e_1 \cdot e_2 \cdot s_2 \cdot s_1$
	$s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot e_1 \cdot s_4 \cdot s_1$	$s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot e_1 \cdot s_1$

TABLE 3.1 – Étude de suivi de trajectoire de N vers D par f_1

$y \in Y^N_{N \xrightarrow{f_1} D}$	s' tel que $s' = P_{N,D}(s)$	Compatibilité	$y \in Y^D_{N \xrightarrow{f_1} D}$
N, M_1, A_2, A_4, B_0	s_1	$s' \in L(H^D)$	N, M_1, A_2, A_3, B_0
	$s_1 \cdot e_1 \cdot s_2 \cdot e_2 \cdot s_1$	$s' \in L(H^D)$	N, M_1, A_2, A_3, B_0
	$s_1 \cdot e_1 \cdot s_1$	$s' \notin L(H^D)$	n'existe pas
N, M_1, A_2, M_4, B_0	$s_1 \cdot e_1 \cdot s_1$	$s' \notin L(H^D)$	"
	$s_1 \cdot e_1 \cdot s_1 \cdot e_1 \cdot s_1$	$s' \notin L(H^D)$	"
N, M_1, M_2, A_4, B_0	s_1, e_1, s_2, s_1	$s' \in L(H^D)$	N, M_1, M_2, A_3, B_0
	$s_1 \cdot e_1 \cdot s_1 \cdot s_2 \cdot s_1$	$s' \notin L(H^D)$	n'existe pas
	$s_1 \cdot e_1 \cdot s_1 \cdot e_1 \cdot s_2 \cdot s_1$	$s' \notin L(H^D)$	"
N, M_1, M_2, M_4, B_0	$s_1 \cdot e_1 \cdot s_1 \cdot e_1 \cdot s_2 \cdot s_1$	$s' \notin L(H^D)$	"
	$s_1 \cdot e_1 \cdot s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot e_1 \cdot e_2 \cdot s_2 \cdot s_1$	$s' \notin L(H^D)$	"
	$s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot e_1 \cdot s_1$	$s' \notin L(H^D)$	"

TABLE 3.2 – Recherche de compatibilité et des états atteignables de N vers D

près la composition du nom des états, nous pouvons en déduire certaines informations. Ainsi, dans ces deux états, le composants C_1 est à l'état M_1. Ceci est raisonnable dans la mesure où l'événement f_1 ne peut justement se produire que lorsque le composant C_1 est dans l'état M_1. Le composant C_2 se trouve être dans deux états différents (A_2 et M_2), mais ce sont des états également présent dans le mode D. Ce n'est donc pas là que le problème se trouve. En effet, en étudiant le comportement du composant C_4, nous nous apercevons que les transitions compatibles sont celles où le composant est dans son état A_4 qui est son état de repos. Les deux transitions incompatibles sont les états M_4 quand le composant est en marche.

Nous pouvons donc en déduire que le système requiert une spécification de commutation liée à l'état du composant non utilisé dans le mode d'arrivée afin que celui-ci se trouve dans un état particulier avant la commutation. Dans notre cas, la spécification de commutation doit imposer que le composant C_4 se trouve dans son état de repos A_4 avant une possible commutation.

Une étude identique dans le sens retour, du mode (dégradé) D vers le mode (nominal) N par

l'événement de commutation r_1 est faite. Nous recherchons comme précédemment si les mots existent dans le langage de l'automate représentant le système dans le mode d'arrivée (mode nominal N). Nous obtenons le tableau 3.3.

$y \in Y^D_{D \xrightarrow{r_1} N}$	$s \in L^y_{D \xrightarrow{r_1} N}(H^D)$	$s' = P_{D,N}(s)$	$y \in Y^N_{D \xrightarrow{r_1} N}$
D,F_1,M_2,A_3,B_0	$s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1$ $s_1 \cdot f_1 \cdot s_3 \cdot e_3 \cdot s_2$	$s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1$ $s_1 \cdot f_1 \cdot s_2$	D,F_1,M_2,A_4,B_0 n'existe pas
D,F_1,A_2,A_3,B_0	$s_1 \cdot f_1$ $s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1 \cdot e_2$	$s_1 \cdot f_1$ $s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1 \cdot e_2$	D,F_1,A_2,A_4,B_0 D,F_1,A_2,A_4,B_0
D,F_1,M_2,M_3,B_0	$s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1 \cdot s_3$ $s_1 \cdot f_1 \cdot s_3 \cdot e_3 \cdot s_2 \cdot s_3$	$s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1$ $s_1 \cdot f_1 \cdot s_2$	D,F_1,M_2,A_4,B_0 n'existe pas
D,F_1,A_2,M_3,B_0	$s_1 \cdot f_1 \cdot s_3$ $s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1 \cdot s_3 \cdot e_2$	$s_1 \cdot f_1$ $s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1 \cdot e_2$	D,F_1,A_2,A_4,B_0 D,F_1,A_2,A_4,B_0
D,F_1,A_2,A_3,B_1	$s_1 \cdot f_1 \cdot s_3 \cdot e_3$ $s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1 \cdot s_3 \cdot e_3 \cdot e_2$	$s_1 \cdot f_1$ $s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1 \cdot e_2$	D,F_1,A_2,A_4,B_0 D,F_1,A_2,A_4,B_0
D,F_1,M_2,A_3,B_1	$s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1 \cdot s_3 \cdot e_3$	$s_1 \cdot e_1 \cdot s_2 \cdot s_1 \cdot f_1$	D,F_1,M_2,A_4,B_0

TABLE 3.3 – Recherche de compatibilité et des états atteignables de D vers N

Une analyse de ce tableau nous montre plusieurs aspects intéressants. Tout d'abord, nous remarquons que tous les langages $L^y_{D \xrightarrow{r_1} N}(H^N)$ sont compatibles, c'est-à-dire qu'ils ont tous au moins *un mot qui existe dans le langage du mode d'arrivée. Par contre, ils ne sont pas* consistants. *En effet, nous remarquons que les langages menant aux états (D,F_1,M_2,A_3,B_0),(D,F_1,M_2,M_3,B_0) et (D,F_1,M_2,A_3,B_1) conduisent tous après projection au même état (D,F_1,M_2,A_4,B_0) du modèle H^N. Il y a la même situation pour les états (D,F_1,A_2,A_3,B_0), (D,F_1,A_2,M_3,B_0) et (D,F_1,A_2,A_3,B_1) qui conduisent au même état (D,F_1,A_2,A_4,B_0). Ceci est un exemple précis de ce qu'est une inconsistance. En effet, une analyse nous montre que lors d'un retour dans le modèle du mode N, le système* ne sait plus *s'il revient dans un état où la machine C_3 est en marche ou à l'arrêt et si le stock est rempli (B_1) ou vide (B_0).*

Deux spécifications s'imposent donc. La première, équivalente à celle énoncée précédemment, requiert que la machine non commune aux deux modes soit dans un état d'arrêt prédéfini (machine C_3 dans l'état A_3). Cette spécification permet d'éviter une commutation alors que la machine C_3 est en cours de production. La deuxième spécification est liée au stock de la machine qui pose problème lors d'une réparation. En effet, il n'existe pas dans le modèle H^N un état qui représente un stock plein et dans lequel arrive un événement de réparation. Le système ne peut être réparé lorsque le stock est plein. Ceci vient du fait que pour que la machine C_1 puisse tomber en panne, il faut qu'elle soit en cours de production. Donc, potentiellement, l'événement de panne f_1 arrive au moment où l'événement incontrôlable de fin de tâche e_1 peut se produire. Or cet événement remplit le stock. Le stock étant déjà rempli, cette fin de tâche aurait provoqué un dépassement de stock, ce qui est interdit par le cahier des charges. Ainsi donc, le composant C_1 ne peut travailler que *si le stock est vide.*

En conclusion, on ne peut réparer le système que *si le stock est vide sous peine d'être incompatible. Cette spécification est difficile à prévoir lors de l'établissement du cahier des charges. Son absence aurait conduit le système vers une erreur lors d'une commutation du système du mode D au mode N dans certaines situations. Cette erreur a été mise en évidence grâce à la recherche de* consistance *entre nos modèles.*

L'ajout des spécifications se fait par un retour à l'étude intermodale 3.6. Nous obtenons ainsi les figures 3.19 représentant le nouveau procédé sous contrôle H^N_{lab} et la figure 3.20 représentant le nouveau procédé sous contrôle H^D_{lab}.

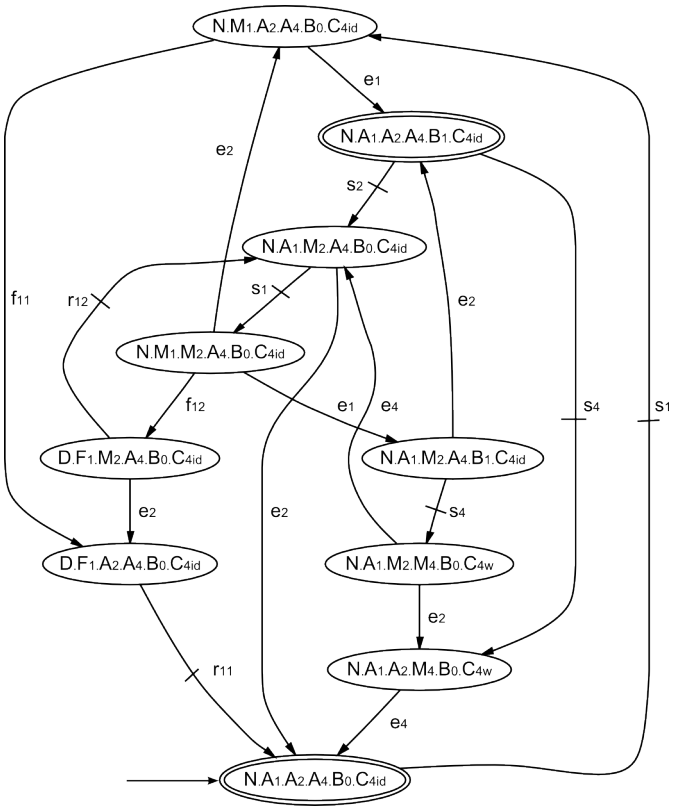

FIGURE 3.19 – Modèle du procédé sous contrôle H^N_{lab} après ajout de la spécification de commutation avec le composant C_4 en état d'arrêt.

Nous pouvons ainsi voir que les commutations incompatibles et non consistantes ont été supprimées, et que les commutations équivalentes dans les deux modèles ont été renommées. Ainsi, sur les deux commutations restantes dans chaque sens, nous avons maintenant les événements f_{11} et f_{12}, représentant une occurrence de l'événement f_1, et les événements r_{11} et r_{12}, représentant l'événement de commutation contrôlable r_1.

À ce stade, les deux modèles sont toujours des procédés sous contrôle. Ils respectent par conséquent les spécifications modélisées. L'étude de suivi de trajectoire assure que les commutations existent et que le système peut commuter d'un mode à l'autre. Enfin la labélisation permet d'anticiper la prochaine phase de fusion d'états provoquant de l'indéterminisme. ⌟

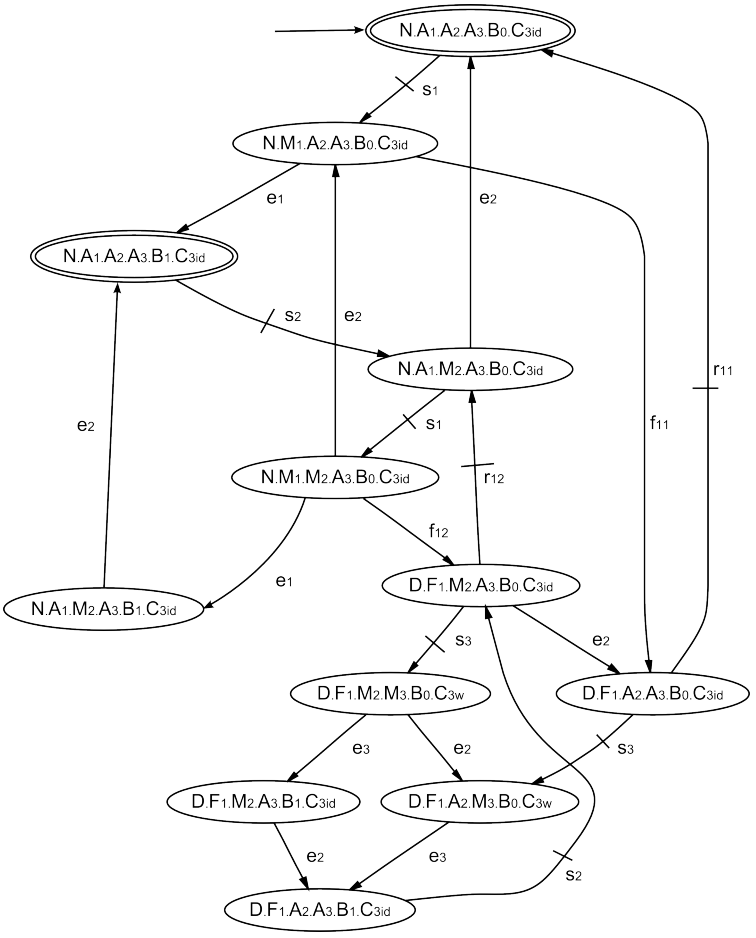

FIGURE 3.20 – Modèle du procédé sous contrôle H^D_{lab} après ajout des spécifications de composants et de stock.

3.8 Fusion des états non significatifs

Dernière étape de notre démarche (présentée par la figure 3.3 à la page 53), la fonction de fusion réduit la complexité des modèles en ne gardant que le comportement interne et le comportement commutatif de chaque mode. Pour obtenir cette dynamique réduite, l'automate de modes $G^{\mathcal{M}}$ a permis d'inclure dans chaque état des modèles l'information du mode actif. Il est ainsi possible d'isoler les états correspondant au comportement interne du système dans chaque mode et de fusionner les autres états. Dans notre nouveau modèle $H^{M_j}_{merge}$ du procédé sous contrôle, l'ensemble des états n'appartenant pas aux modes, appelés *états non significatifs*, forme un nouvel état appelé *état inactif*, nommé $y^{M_j}_{id}$, représentant l'inactivité du mode. Cet état permet de représenter l'unicité de mode actif, c'est-à-dire que pour N modes, $N-1$ modes seront dans cet état d'inactivité et seul un mode n'y sera pas. Ce sera ce dernier qui représentera le comportement du système.

Nous désirons également que les modèles construits aient un comportement concurrent entre eux. C'est à dire que ce n'est pas qu'un seul modèle qui représente le comportement du système, mais l'ensemble de leurs comportements qui représente le comportement global du système. Vouloir un comportement concurrent entre modèles revient à effectuer une composition parallèle implicite entre eux.

Ceci peut être exprimé par la définition suivante :

Définition 39 (Comportement global équivalent)
En considérant les modèles H_{merge} représentant le comportement concurrent de tous les modes du système, avec $\mathcal{M}$ l'ensemble des modes. Nous avons : $H_{merge} = ||_{M_j \in \mathcal{M}} H^{m_j}_{merge}$ ♦

Or la composition parallèle synchronise les événements communs. Il faut donc s'assurer, lorsque les $N-1$ modes se trouvent dans leur état inactif, qu'ils ne vont pas restreindre le comportement du mode qui représente la part du comportement actuel du système. C'est pour cela qu'il faut s'assurer que l'état inactif peut générer tous les événements possibles, soit en boucle, soit pour sortir de cet état.

Formellement, si $Y^{M_j}_{mer}$ est l'ensemble des états non significatifs, l'automate $H^{M_j}_{merge}$ est défini tel que :

Définition 40 ($H^{M_j}_{merge}$: Procédé sous contrôle fusionné dans le mode M_j)
$H^{M_j}_{merge} = (Y^{M_j}_{merge}, \Sigma^{M_j}_{merge}, \tau^{M_j}_{merge}, y^{M_j}_{merge,0}, Y^{M_j}_{merge,m})$ *tel que :*
- $Y^{M_j}_{merge} = (Y^{M_j}_{lab} \setminus Y^{M_j}_{mer}) \cup \{y^{M_j}_{id}\}$

- $\Sigma^{M_j}_{merge} = \Sigma^{M_j}_{lab} \setminus \bigcup_{C_i \in (C^{M_j}_{\rightleftarrows} \setminus C^{M_j}_{\circlearrowleft})} \Sigma^{C_i}_{\circlearrowleft}$
- $\tau^{M_j}_{merge} = \begin{array}{l} \tau^{M_j}_{lab} \quad \setminus \{((y_a,s),y_b) \mid s \in \Sigma^{M_j}_{lab},\ y_a,y_b \in Y^{M_j}_{mer},\ \tau^{M_j}_{lab}(y_a,s) = y_b \text{ est défini}\} \\ \qquad \cup \{((y^{M_j}_{id},s'),y^{M_j}_{id}) \mid s' \in (\Sigma^{M_j} \setminus \Sigma^{M_j}_{\leftarrow})\} \end{array}$
- $y^{M_j}_{merge,0} = \begin{cases} y^{M_j}_{lab,0} & \text{si} \quad y^{M_j}_{lab,0} \notin Y^{M_j}_{mer} \\ y^{M_j}_{id} & \text{sinon} \end{cases}$
- $Y^{M_j}_{merge,m} = \begin{cases} Y^{M_j}_{lab,m} & \text{si} \quad Y^{M_j}_{lab,m} \cap Y^{M_j}_{mer} = \emptyset \\ Y^{M_j}_{lab,m} \setminus Y^{M_j}_{mer} \cup \{y^{M_j}_{id}\} & \text{sinon} \end{cases}$ ♦

Remarque 5

Du point de vue de la fonction de transition $\tau^{M_j}_{merge}$, celle-ci s'exprime par toutes les transitions présentes dans le modèle $H^{M_j}_{lab}$ duquel nous enlevons toutes les transitions qui partent d'un état appartenant à l'ensemble des états à fusionner $Y^{M_j}_{mer}$ et qui vont vers un état appartenant également à $Y^{M_j}_{mer}$. Ensuite, pour permettre un comportement concurrent tel qu'exprimé par la définition 39, nous ajoutons en boucle sur l'état inactif $y^{M_j}_{id}$ une transition par événement de l'alphabet du mode, hormis pour les événements de commutations entrant dans ce mode, c'est-à-dire sortant de l'état $y^{M_j}_{id}$. ♦

Nous proposons la procédure suivante pour construire le modèle $H^{M_j}_{merge}$.

Procédure 3

En considérant les automates $H^{M_j}_{lab}$ et $H^{M_j}_{merge}$ tel que $H^{M_j}_{merge}$ soit le modèle obtenu par fusion d'états de l'automate $H^{M_j}_{lab}$, la procédure de fusion d'états est décrite comme suit :

1. *nous déterminons dans $H^{M_j}_{lab}$, un ensemble d'état à fusionner $Y^{M_j}_{mer} \subset Y^{M_j}_{lab}$. Les états inclus dans $Y^{M_j}_{mer}$ sont les états non-significatifs du mode M_j dont le nom n'est pas composé par M_j (cette particularité est donnée par l'automate de modes $G^{\mathcal{M}}$ lors de l'étape d'extension des modèles) ;*
2. *tous les états de $Y^{M_j}_{mer}$ sont remplacés par un nouvel état appelé $y^{M_j}_{id}$;*
3. *si l'état initial est inclus dans $Y^{M_j}_{mer}$, alors $y^{M_j}_{id}$ est le nouvel état initial. Sinon, l'état initial reste celui du modèle $H^{M_j}_{lab}$;*
4. *si au moins un état marqué est inclus dans $Y^{M_j}_{mer}$, alors $y^{M_j}_{id}$ est un état marqué.* ♦

À la suite de cette fonction de fusion, le concepteur possède un modèle par mode qui représente le comportement interne du système dans le mode considéré et dans lequel les événements de commutation activent ou désactivent le mode en le faisant entrer ou sortir de son état inactif. Ces comportements distincts respectent les contraintes liées à chaque mode et le concepteur est

en mesure d'extraire la loi de commande à implémenter.

⌜ *Exemple*

Comme expliqué dans cette section, les modèles H^N_{lab} et H^D_{lab} contiennent trop d'informations au regard de ce dont nous avons besoin. Cette étape de fusion supprime l'information précédemment nécessaire pour assurer une recherche de dynamique respectant les contraintes et les commutations existantes. Cette information est maintenant non-significative et peut être enlevée afin d'améliorer l'interprétation des modèles. Ainsi, dans le modèles H^N_{lab}, illustré figure 3.21 (figure du haut), les états non-significatifs sont les états dans lesquels le système est dans le mode D. Ce sont les états (D,F_1,A_2,A_4,B_0,C_{4id}) et (D,F_1,M_2,A_4,B_0,C_{4id}). Ces états vont fusionner afin de devenir l'état inactif q^N_{id} de notre mode. Le nouveau modèle H^N_{merge} est illustré figure 3.21 (en bas).

La même opération est réalisée sur le modèle H^D_{lab}. Nous obtenons le modèle H^D_{merge} illustré figure 3.22. Le gain de taille est beaucoup plus grand dans ce modèle. En effet, le mode D nécessite beaucoup d'informations afin d'assurer une fiabilité de commutation.

Nous pouvons visualiser également sur les deux figures l'importance de la phase de labélisation à la fin de l'étude de suivi de trajectoire. Les trajectoires sortantes des états inactifs q^N_{id} et q^D_{id} auraient le même nom si nous n'avions pas labélisé ces commutations avant l'étape de fusion.

De ces deux modèles, le concepteur peut maintenant extraire les lois de commande de son système. ⌟

3.9 Comparaison avec l'approche centralisée

L'objectif de cette section est de présenter une comparaison formelle entre l'approche centralisée couramment utilisée dans la Théorie de Contrôle par Supervision et l'approche modale utilisée dans notre démarche.

Pour effectuer cette comparaison, nous nous plaçons volontairement dans un cas où le concepteur a un point de vue orienté gestion de modes sur les modèles. Autrement dit, même par une approche centralisée, le concepteur souhaite faire ressortir des modes de fonctionnement et par conséquent tient compte dans l'approche centralisée des comportements de mode. Ainsi, l'automate de modes $G^{\mathcal{M}}$ est inclus dans le comportement du procédé G dans l'approche centralisée.

De plus, pour l'étude de cette comparaison, nous nous plaçons dans un cas où les modèles sont consistants entre eux (cf. définition 38). Si tel n'est pas le cas, alors il n'est pas possible de respecter les spécifications dans l'approche centralisée comme dans l'approche modale. De même, l'ajout de spécifications suite à des modèles non consistants n'est qu'une étape permettant d'identifier un problème et de le corriger. Sans ces spécifications, l'approche centralisée

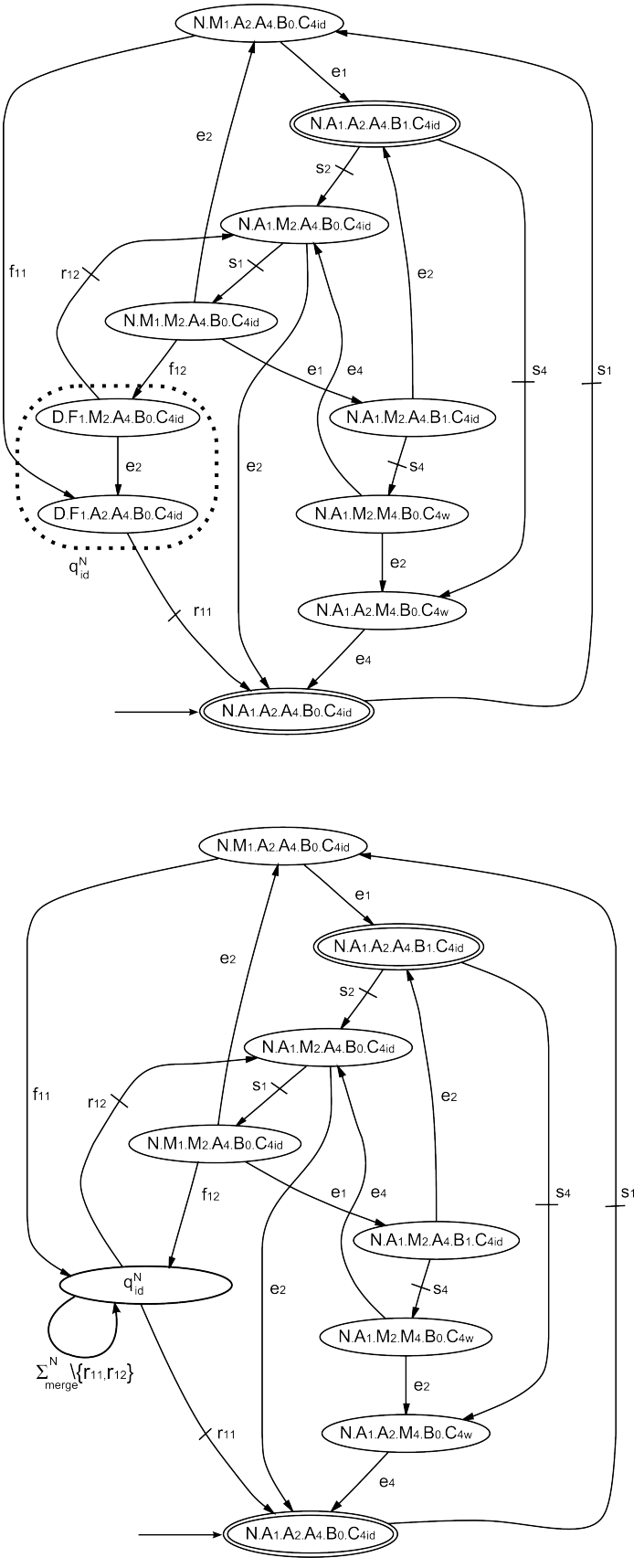

FIGURE 3.21 – Modèle H^N_{lab} (haut) et modèle H^N_{merge} (bas)

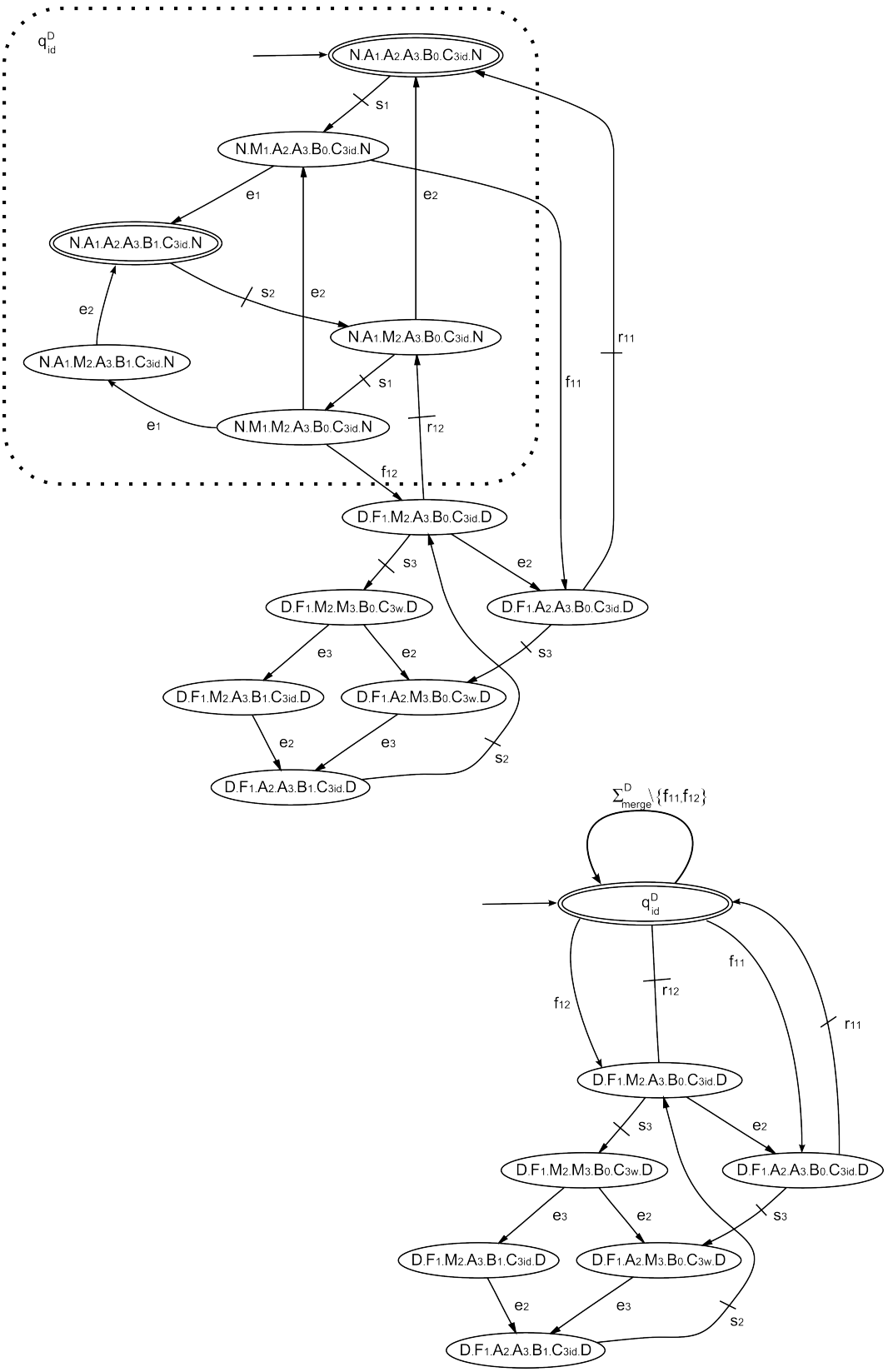

FIGURE 3.22 – Modèle H^D_{lab} (haut) et modèle H^D_{merge} (bas)

aurait conduit à un comportement non spécifié, mais non proscrit par la spécification. C'est le cas lorsqu'un comportement n'est pas assez précisé dans le cahier des charges. L'approche centralisée considère comme permis, donc correct, tout ce qui n'est pas interdit, alors que l'approche modale indique un problème potentiel [5].

Nous considérons également que la labélisation d'événements n'a pas été faite durant la procédure de suivi de trajectoire (étape 1(c)iiB de la procédure 2). Si nous considérons cela, c'est pour trois raisons. La première est qu'il est impossible de comparer des modèles qui ne sont pas exprimés sur le même ensemble d'événements : les événements labélisés n'existent pas dans le modèle du procédé sous contrôle construit par l'approche centralisée. La seconde est que la labélisation des événements n'est utilisée que pour lever l'indéterminisme provoqué par la fusion d'états à l'étape suivante de la démarche. Elle n'apporte aucune information pour valider la consistance. De plus, elle est réalisée à ce stade par commodité. En effet, le nouvel événement labélisé correspond à une trajectoire unique. L'indice rajouté à l'événement sert donc à discriminer une trace précise. D'un point de vue implémentation, l'événement labélisé reste ce qu'il était avant labélisation, mais un signal unique, représenté par l'événement et par son indice, est produit lorsque la trace qu'il identifie est générée. C'est pourquoi nous ne tenons pas compte de la labélisation pour l'équivalence des modèles. Enfin, la troisième est liée à la fonction de fusion qui regroupe les états non-significatifs pour le mode considéré, comme expliqué dans la section 3.8. La fusion d'états provoque de l'indéterminisme. Cet indéterminisme était levé grâce à la labélisation effectuée à l'étape précédente. Or, nous considérons que la labélisation n'est pas faite pour éviter de comparer des modèles qui ne sont pas exprimés sur le même ensemble d'événements. Par conséquent, il est inutile de considérer les modèles avec les états fusionnés.

Ce sont les raisons pour lesquelles nous effectuons cette comparaison sur les modèles H^{M_j}, en nous assurant au préalable qu'ils soient consistants entre eux.

Notre objectif pour cette comparaison est de démontrer une équivalence de comportement entre le modèle du procédé sous contrôle H_{cent}, construit par l'approche centralisée, et les modèles du procédé sous contrôle H^{M_j}, construits par l'approche modale, lorsque ceux-ci fonctionnement de manière concurrente.

Notre objectif est donc de vérifier l'égalité suivante :

$$H_{cent} = ||_{M_j \in \mathcal{M}} H^{M_j} \tag{3.1}$$

Remarque 6

Que ce soit dans l'approche centralisée ou dans l'approche modale, nous utilisons systémati-

5. Prenons l'exemple d'une machine y en suppléance d'une machine x dans le cas où cette dernière tombe en panne. Si le concepteur n'indique pas, après la réparation de la machine x, si la machine y doit s'arrêter avant de relancer la machine x, alors les deux machines fonctionneront pendant un laps de temps. C'est un comportement non spécifié par le concepteur et non proscrit par des spécifications. Ce cas là est identifié par l'approche modale et est signalé au concepteur pour qu'il puisse spécifier si besoin.

quement par commodité la composition parallèle au lieu du produit. La raison est qu'en considérant deux automates avec le même ensemble d'événements, la composition parallèle et le produit donnent un résultat identique (cf. définitions 11 et 12). Pour éviter de mélanger dans une même égalité l'opérateur de composition parallèle et le produit, nous prendrons comme seul opérateur la composition parallèle. ♦

3.9.1 Rappel de construction des modèles

Nous commençons par rappeler brièvement les étapes de constructions des modèles de procédé sous contrôle pour les deux approches.

Approche centralisée

Dans l'approche centralisée, et dans un cas classique d'utilisation de la Théorie de Contrôle par Supervision, le concepteur construit son procédé G par l'ensemble des modèles représentant ces composants, et son modèle de spécification E par l'ensemble des modèles représentant chaque spécification séparément. De plus, il est rajouté le comportement de l'automate de modes $G^{\mathcal{M}}$ pour y inclure le comportement modal.

Pour le procédé G, en considérant l'ensemble des composants $\mathcal{C}$, G^{C_i} les modèles de composant C_i et $G^{\mathcal{M}}$ l'automate de modes, le procédé G_{cent} est défini tel que :

$$G_{cent} = G^{\mathcal{M}} || (||_{C_i \in \mathcal{C}} G^{C_i}) \tag{3.2}$$

Pour le modèle des spécifications E, en considérant le modèle E^l, avec $l \in \mathbb{N}$, représentant une spécification élémentaire, alors le modèle des spécification est défini par :

$$E = ||_l E^l \tag{3.3}$$

Le procédé sous contrôle se construit alors par la composition parallèle de ces deux modèles, tel que :

$$H_{cent} = G_{cent} || E. \tag{3.4}$$

Dans le cas où la propriété de contrôlabilité n'est pas respectée (cf. théorème 2), le procédé sous contrôle génère le langage suprême contrôlable (cf. section 1.3.3) : $L_m(H_{cent}) = [L_m(G_{cent} \times E)]^{\uparrow c}$.

Approche modale

Dans l'approche modale, plusieurs modèles H^{M_j} – un par mode – de procédé sous contrôle sont construits. Ces modèles sont construits à partir des modèles des procédés et des spécifications dans les modes considérés.

Les modèles G^{M_j} représentent les procédés dans chaque mode. Leur construction est définie déf. 29 page 64 telle que :

$$G^{M_j} = [||_{C_k \in \mathcal{C}^{M_j}} G^{C_k}]||G^{\mathcal{M}} \tag{3.5}$$

avec $G^{\mathcal{M}}$ l'automate de modes défini déf. 26 page 56.

Pour les modèles des spécifications E^{M_j}, leur construction est définie déf. 30 page 68 telle que :

$$E^{M_j} = ||_k E^{k,M_j} \tag{3.6}$$

Les modèles H^{M_j} représentant le procédé sous contrôle dans le mode M_j se construisent par la composition parallèle des modèles du procédé et des spécifications dans chaque mode, comme défini déf. 31 page 70 , tels que :

$$H^{M_j} = G^{M_j}|| \, E^{M_j} \tag{3.7}$$

Dans le cas où la propriété de contrôlabilité n'est pas respectée, le procédé sous contrôle dans le mode est le langage suprême contrôlable : $L_m(H^{M_j}) = [L_m(G^{M_j} \times E^{M_j})]^{\uparrow c}$

3.9.2 Équivalence des procédés sous contrôle

Dans l'approche modale, les procédés sous contrôle H^{M_j} représentent une part du comportement du système. Chacun de ces procédés sous contrôle est activé séquentiellement, et un seul est actif à la fois. Les autres sont non actifs au sens où ils se trouvent dans leur état inactif (voir section 3.8).

Notre objectif pour valider l'équivalence de comportement est de respecter l'égalité 3.1.

Or, d'après les égalités 3.4 et 3.7, l'égalité 3.1 peut s'écrire sous la forme suivante :

$$G_{cent}||E = ||_{M_j \in \mathcal{M}}(G^{M_j}||E^{M_j}) \tag{3.8}$$

De même, d'après les propriétés de commutativité et d'associativité de la composition parallèle, l'égalité 3.8 peut se mettre sous la forme :

$$G_{cent}||E = (||_{M_j \in \mathcal{M}} G^{M_j})||(||_{M_j \in \mathcal{M}} E^{M_j}) \tag{3.9}$$

Au vue de la forme de l'égalité 3.9, nous pouvons en déduire qu'il y a équivalence des procédés sous contrôle au sens de l'égalité 3.1 si les égalités suivantes sont vérifiées :

$$G_{cent} = ||_{M_j \in \mathcal{M}} G^{M_j} \tag{3.10}$$

$$E = ||_{M_j \in \mathcal{M}} E^{M_j} \tag{3.11}$$

L'égalité 3.10 fait référence à l'équivalence des modèles du procédé et signifie que le comportement résultant de la composition parallèle des procédés de l'approche modale est égal au comportement du procédé construit par l'approche centralisée. De même, l'égalité 3.11 fait référence aux modèles des spécifications et implique que le comportement résultant de la composition parallèle des spécifications de l'approche modale soit égal au comportement de la spécification construite dans l'approche centralisée. Si ces deux égalités sont respectées, alors le comportement de la composition parallèle des procédés sous contrôle dans l'approche modale est identique au comportement du procédé sous contrôle construit par l'approche centralisée.

Les sections suivantes présentent les conditions nécessaires pour respecter ces hypothèses.

3.9.3 Équivalence des procédés

Dans cette section, nous présentons les conditions pour que l'égalité 3.10 soit vraie et que la composition parallèle des procédés de l'approche modale donne un comportement identique à celui du procédé construit par l'approche centralisée.

Le point de départ est l'égalité 3.10 qui est $G_{cent} = ||_{M_j \in \mathcal{C}} G^{M_j}$.

Les procédés construits par l'approche centralisée sont décrits par l'égalité 3.2. De même, les procédés construits par l'approche modale sont définis par l'égalité 3.5.

À partir des éléments ci-dessus, et en considérant les propriétés de la composition parallèle, il est maintenant possible de vérifier l'égalité 3.10.

Proposition 1

$$
\begin{aligned}
& G^{M_j} = G^{\mathcal{M}} || (||_{C_k \in \mathcal{C}^{M_j}} G^{C_k}) \\
\Leftrightarrow \quad & ||_{M_j \in \mathcal{M}} G^{M_j} = ||_{M_j \in \mathcal{M}} (G^{\mathcal{M}} || (||_{C_k \in \mathcal{C}^{M_j}} G^{C_k})) \\
\Leftrightarrow \quad & ||_{M_j \in \mathcal{M}} G^{M_j} = (||_{M_j \in \mathcal{M}} G^{\mathcal{M}}) || (||_{M_j \in \mathcal{M}} (||_{C_k \in \mathcal{C}^{M_j}} G^{C_k}))
\end{aligned}
$$

mais, sachant que

$$
\begin{aligned}
||_{M_j \in \mathcal{M}} G^{\mathcal{M}} &= ||^n G^{\mathcal{M}} \\
&= (\underbrace{G^{\mathcal{M}} || G^{\mathcal{M}} || \ldots || G^{\mathcal{M}}}_{\text{n fois}}) \\
&= G^{\mathcal{M}}
\end{aligned}
$$

et que $||_{M_j \in \mathcal{M}} (||_{C_k \in \mathcal{C}^{M_j}} G^{C_k}) = ||_{C_k \in \mathcal{C}} G^{C_k}$

par conséquent

$||_{M_j \in \mathcal{M}} G^{M_j} = G^{\mathcal{M}} || (||_{C_k \in \mathcal{C}} G^{C_k})$

$||_{M_j \in \mathcal{M}} G^{M_j} = G$

Cette dernière égalité correspond bien à l'hypothèse exprimée par l'égalité 3.10. Cette proposition montre que si tous les composants d'un système, utilisés pour construire le procédé de l'approche centralisée, appartiennent bien tous à *au moins* un mode, alors, en considérant que le procédé dans un mode soit la composition parallèle des composants utilisés par le système dans ce mode, le comportement généré par la composition parallèle des procédés dans les modes du système est identique à celui du procédé construit par l'approche centralisée.

La vérification que tous les composants appartiennent à *au moins* un mode correspond à

l'étape de la cohérence des modèles de notre démarche et est exprimée section 3.4 et plus particulièrement définitions 25 et 27.

3.9.4 Équivalence des spécifications

Dans cette section, nous présentons les conditions pour que l'égalité 3.11 soit vraie et que la composition parallèle des modèles de spécifications utilisés dans l'approche modale donne un comportement identique au modèle de spécification utilisé dans l'approche centralisée.

Le point de départ est l'égalité 3.11 qui est $E = ||_{M_j \in \mathcal{M}} E^{M_j}$.

Pour la suite, nous nous basons sur les égalités 3.3 et 3.6. Nous faisons également l'hypothèse qu'il n'y a qu'une seule spécification élémentaire. Autrement dit, dans les indices l et k dans les égalités 3.3 et 3.6 sont égales à 1. Ainsi, nous cherchons à démontrer directement l'égalité 3.11. Cependant, nous garderons l'indice dans nos égalités pour éviter toutes conclusions hâtives.

Pour cela, nous faisons une démonstration en deux parties. La première suppose que la spécification E^{M_j} représentée dans les modes est identique quel que soit le mode considéré. C'est-à-dire que le comportement représenté par cette spécification implique toujours les mêmes événements et n'agit pas sur les autres. Ceci peut être le cas dans certains cas tel qu'une spécification sécuritaires à vérifier, un stock utilisé dans tous les modes, etc.

La deuxième partie considère que les modèles représentant une spécification élémentaire dans chaque mode, est différente suivant le mode considéré. Ce cas, opposé au premier, est habituel le cas dans la gestion des modes, notamment pour ce qui concerne les spécifications de commutations, qui sont propres à chaque mode.

Équivalence : spécification élémentaire indépendante du mode

Dans le cas où la spécification élémentaire est représentée identiquement quel que soit le mode, à un alphabet près, alors nous pensons que son comportement influence de manière identique le comportement construit par approche centralisée comme le comportement équivalent construit par approche modale.

Proposition 2

$$\begin{aligned} & E = ||_{M_j \in \mathcal{M}} E^{M_j} \\ \Leftrightarrow \quad & ||_l E^l = ||_{M_j \in \mathcal{M}} (||_k E^{k,M_j}) \\ \Leftrightarrow \quad & ||_l E^l = ||_k (||_{M_j \in \mathcal{M}} E^{k,M_j}) \end{aligned}$$

mais, sachant que
$$\begin{aligned} ||_{M_j \in \mathcal{M}} E^{k,M_j} &= ||^n E^{k,M_j} \text{ car spécification indépendante} \\ &= (\underbrace{E^{k,M_j} || E^{k,M_j} || \ldots || E^{k,M_j}}_{\text{n fois}}) \\ &= E^{k,M_j} \end{aligned}$$

et que $||_k (||_{M_j \in \mathcal{M}} E^{k,M_j}) = ||_k E^k$

par conséquent, Si E^l est une spécification élémentaire et s'écrit indépendant du mode considéré, alors :
$E = ||_{M_j \in \mathcal{M}} E^{M_j}$

Équivalence : spécification élémentaire dépendante du mode

Le cas présent est celui où les spécifications sont propres à chaque mode. Plus particulièrement encore en considérant l'exemple où une spécification élémentaire n'existe que dans quelques modes. Dans ce cas, son modèle dans ce mode est simple. Elle commence à contraindre le comportement du mode dès que ce mode s'active et jusqu'à ce que ce mode se désactive indépendamment des autres modes, vues qu'ils ne sont pas représentés dans l'approche modale. En revanche, construire son modèle dans l'approche centralisée est plus difficile car il demande à prendre en compte les spécifications des autres modes pour être sûr qu'elle ne va pas restreindre plus qu'elle ne devrait.

À ce niveau, on ne peut faire d'hypothèse. Il nous faut alors bien valider l'égalité :
$E^l = ||_{M_j \in \mathcal{M}} E^{k,M_j}$

La seule manière de respecter une équivalence de comportement est donc que la composition parallèle des modèles représentant la spécification élémentaire dans chaque modes soit égal au comportement de la même spécification décrite par un unique modèle dans l'approche centralisée.

Si tel n'est pas le cas, alors deux modèles différents qui génèrent deux langages différents peuvent improbablement donner un procédé sous contrôle identique.

3.9.5 Synthèse

Après étude des égalités et propositions de cette section, nous proposons la définition suivantes concernant l'équivalence de modèles.

Définition 41 (Équivalence de comportement entre modèles)

Condition 6
Le comportement du procédé construit par approche centralisée est équivalant à la composition parallèle des procédés construits par approche modale si : tous les composants du système sont bien utilisés dans au moins *un mode* M_j *:* $\mathcal{C} = \bigcup_{M_j \in \mathcal{M}} \mathcal{C}^{M_j}$ ♦

Condition 7
Le modèle des spécifications construit pour l'approche centralisée est équivalant à la composition parallèle des modèles des spécifications par approche modale si le modèle de la spécifica-

tion est indépendant du mode considéré : $||_k E^k = ||_k(||_{M_j \in \mathcal{M}} E^{k,M_j})$ *;* ♦

Dans le cas où la condition 7 de la définition 41 n'est pas vérifiée, alors l'équivalence de comportement n'est pas assuré. Le comportement équivalent des procédés sous contrôle de l'approche modale peut alors être plus réduit que le comportement du procédé sous contrôle de l'approche centralisée.

3.10 Conclusion

Dans ce chapitre, nous avons présenté une démarche de conception contribuant à valider une approche modale des SED. Notre démarche est décomposée en étapes, répondant successivement à la modélisation de modes, aux états entrants et sortants de mode, et enfin aux commutations.

La première étape se focalise sur la formalisation des exigences. Celle-ci est cruciale car les modèles qui en sont issus sont les premières données manipulables à ce stade de la conception. Si une erreur se produit à ce niveau, elle se répercute dans toute la conception. La formalisation, à travers des définitions et des conditions à respecter, assure qu'aucune faute de modélisation grossière n'est apparue.

La deuxième étape de la démarche étudie localement les comportements internes du système dans chaque mode. Ces études locales permettent de diminuer la taille des modèles manipulés et ainsi d'avoir une meilleure interprétation de ceux-ci. La conséquence principale est d'améliorer la modélisation des spécifications qui sont ensuite utilisées dans des modèles de taille plus importante.

La troisième étape concerne l'admissibilité des commutations dans les modes. Dans cette étape, nous prenons en compte par extension des modèles de toutes les commutations possibles. Ensuite, nous nous assurons du respect des spécifications de commutations par une recherche de contrôlabilité. Cela permet de certifier qu'aucune commutation non désirée dans un mode ne puisse se produire.

La quatrième étape est relative aux commutations entre modes. Le problème consiste à s'assurer de l'admissibilité des commutations à l'égard des spécifications des modes destinataires. L'admissibilité est recherchée par une procédure de suivi de trajectoire qui évalue la cohérence des commutations et la consistance des modèles de modes. Dans le cas où la consistance n'est pas vérifiée, la procédure indique les trajectoires problématiques pour aider dans leur résolution.

Ces propositions s'inscrivent dans la continuité des travaux de thèse précédents, L'approche multi-modèle préconisée se perçoit comme la recherche d'une validation de comportements concurrents. Cependant, pour avoir un modèle propre à chaque mode, contenant seulement les comportements interne et commutatif, et pour permettre un comportement concurrent entre modèles, il est procédé en fin de démarche à une fusion des états non significatifs qui ne repré-

sentent pas le comportement propre à un mode. Nous obtenons alors un comportement concurrent équivalent au comportement d'un modèle monolithique du système. Cette équivalence de comportement est démontrée, et est vraie notamment dans le cas où l'approche centralisée et l'approche modale ont les mêmes spécifications.

Notre démarche permet ainsi, à travers cinq étapes, de partir d'un cahier des charges et de fournir les modèles de modes, en passant par l'étude des comportements interne et commutatif, et d'étudier les commutations entre modes. Elle utilise les concepts précédemment développés tels que l'ajout d'un état inactif, et la nécessité d'effectuer un suivi de trajectoire pour éviter la perte d'information lors d'un changement de mode. De plus, elle comble également les manques de ces travaux antérieurs qui se caractérisaient par une démarche incomplètement définie, dans l'ajout manuel des commutations sans prise en compte des spécifications sur ces commutations.

4

Application à un système industriel

Sommaire

4.1 Introduction

L'objectif de ce chapitre est de montrer l'applicabilité de la démarche proposée sur un exemple industriel de taille importante. En effet, la mise à l'échelle sur des systèmes physiques réels est un problème récurrent pour les propositions académiques. L'exemple retenu est un système de fabrication flexible présenté une première fois dans les travaux de [Que05] puis repris dans différents travaux [Sch07, Hil08, Hil10]

Cependant, cet exemple ne comporte initialement aucun mode de fonctionnement mais possède un seul fonctionnement permanent. Dans ce fonctionnement, plusieurs stratégies peuvent être mises en oeuvre suivant les objectifs à atteindre. Nous avons donc légèrement modifié le cahier des charges initial afin de faire apparaître des modes de fonctionnement.

La prochaine section est consacrée à la présentation du système étudié et des spécifications attendues. Les suivantes sont les différentes étapes de la démarche telle qu'illustrée par la figure 3.3 à la page 53.

4.2 Présentation de l'exemple

Le système flexible à étudier est illustré par la figure 4.1. Il s'agit d 'un système de fabrication et d'assemblage de deux types de goupilles, cylindrique ou conique peinte, à partir d'une pièce support et d'une cheville.

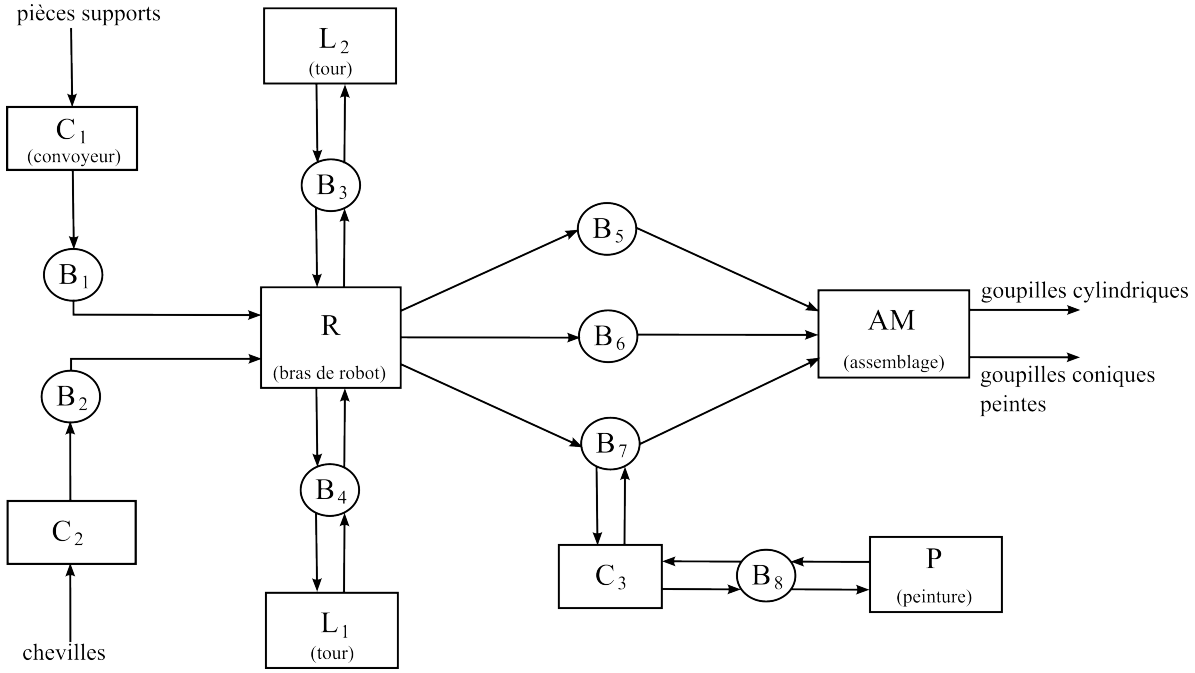

FIGURE 4.1 – Système de fabrication flexible étudié.

4.2.1 Description du système

Le système étudié comporte 8 éléments : 3 convoyeurs (C_1,C_2 et C_3), 2 tours (L_1,L_2), un robot (R), une unité de peinture (P) et une unité d'assemblage (AM).

L'objectif de production est d'élaborer 2 types d'assemblage réalisés à partir d'un support et d'une cheville ayant tous deux subis des opérations spécifiques. Entre chaque opération ou manipulation, les pièces seront déposées dans des stocks intermédiaires identifiés (B_1,...,B_8) de capacité maximale d'une unité. Avant toute opération, la pièce support est transférée de manière autonome par le convoyeur C_1 puis déposée dans B_1, et respectivement une cheville transférée par le convoyeur C_2 puis déposée dans B_2. Toutes les autres manipulations sont réalisées par le robot exception faite du stock B_8 qui sera chargé/déchargé par ses 2 éléments contigus (P et C_3). En début d'opération, le robot prélèvera une pièce support de B_1 pour la déposer soit en B_3 ou B_4 (noté $B_{3,4}$), respectivement il prélèvera une cheville de B_2 pour la déposer en $B_{3,4}$. La pièce support d'assemblage sera usinée par l'un des tours L_1 ou L_2 (noté $L_{1,2}$) avant d'être déchargée en B_5. Suivant le type d'assemblage, les chevilles suivront 2 opérations distinctes. Pour le premier type (type 1), un usinage de base peut être effectué par l'un des tours $L_{1,2}$. La pièce ainsi traitée sera déposée en B_6. Pour le second type (type 2), un usinage plus poussé est nécessaire et ne peut être effectué que par $L1$. La cheville ainsi pré-traitée est déposée en B_7 pour être ensuite peinte par P puis redéposée en B_7. L'unité de peinture opère sur des chevilles stockées en B_8 et transférées par C_3. Les assemblages s'effectuent alors à partir d'une pièce support pré-traitée localisée en B_5 avec une cheville préalablement traitée et localisée soit en B_6, soit en B_7.

Pour éclaircir ce fonctionnement, nous proposons le tableau suivant qui récapitule la trajet, à travers les éléments du système, des pièces en entrées afin de fabriquer les pièces de sorties.

Pièces en entrées	éléments traversés par les pièces
Pièces supports	$C_1 \rightarrow B_1 \rightarrow R \rightarrow B_{3,4} \rightarrow L_{1,2} \rightarrow B_{3,4} \rightarrow R \rightarrow B_5$
Chevilles	$C_2 \rightarrow B_2 \rightarrow R \rightarrow B_{3,4} \rightarrow L_{1,2} \rightarrow B_{3,4} \rightarrow R \rightarrow B_6$
	$C_1 \rightarrow B_1 \rightarrow R \rightarrow B_4 \rightarrow L_1 \rightarrow B_4 \rightarrow R \rightarrow B_7 \rightarrow C_3 \rightarrow B_8 \rightarrow P \rightarrow B_8 \rightarrow C_3 \rightarrow B_7$

À partir des pièces contenues dans les stocks B_5,B_6 et B_7, nous sommes en mesure de fabriquer les pièces de sorties :

Pièces désirées en sortie	Pièces nécessaires à l'assemblage et provenance
Goupilles cylindriques	1 pièce de B_5 + 1 pièce de B_6
Goupilles coniques peintes	1 pièce de B_5 + 1 pièce de B_7

Nous voyons alors que quel que soit le type de goupille en sortie, il est nécessaire d'avoir une pièce support stockée dans B_5. Ce qui détermine alors le type de la goupille en sortie est le traitement effectué sur la cheville en entrée.

4.2.2 Cahier des charges

Ce cahier des charges exprime les différents comportements attendus du système, ainsi que les objectifs à atteindre.

Le système est exploité pour répondre à une demande de production. Il n'est pas utilisé pour la production simultanée des deux types de goupilles. Nous souhaitons passer d'une production à l'autre suivant un requête venant du pupitre de commande (noté *PR*). L'événement *avp* correspond à une demande d'activation de l'unité de peinture pour la fabrication de goupilles coniques, alors que l'événement *ssp* représente la désactivation de cette unité, donc un traitement normal, pour la fabrication de goupilles cylindriques.

Le tour principal L_1, qui traite toutes les pièces suivant la production demandée, peut cependant tomber en panne. Cette panne est signalée par l'événement f_1 lorsque la fin de tâche ne se réalise pas convenablement. Un technicien est alors en charge de réparer la machine et autorise en fin de réparation la remise en marche de celle-ci par l'événement r_1.

Pour ne pas arrêter la production en cours, le système dispose d'un tour de remplacement L_2. Ce tour prendra en charge l'usinage de L_1 lorsque ce dernier est en panne. Néanmoins, le tour L_2 n'est pas une machine identique à L_1. Elle est moins performante et permet moins d'opérations. C'est la raison pour laquelle le tour L_2 est incapable d'effectuer le traitement sur les chevilles permettant de produire des goupilles coniques peintes. Par conséquent, la panne du tour L_1 et l'activation du tour L_2 implique la fabrication exclusive, le temps de la réparation, de goupilles cylindriques.

À l'étude des précédents paragraphes, il y a *au moins* trois modes de fonctionnement : deux modes nominaux, correspondant à la fabrication de goupilles cylindriques ou de goupilles coniques peintes, et un mode dégradé représentant la panne du tour L_1 et l'utilisation du tour L_2 pour la fabrication de goupilles cylindriques.

Concernant le bras de robot, celui-ci possède deux configurations particulières dépendant de l'utilisation du stock B_3 ou du stock B_4. En effet, il ne peut utiliser les deux stocks en même temps. En revanche, il peut accéder aux autres stocks indépendamment de sa configuration. Ainsi, suivant l'utilisation du stock B_3 ou B_4, le bras de robot doit effectuer une reconfiguration pour être en mesure de déposer/prendre une pièce dans l'un ou l'autre des stocks. Néanmoins, avant d'autoriser une reconfiguration, les stocks B_3 et B_4 soient vides afin de ne pas garder de pièces inutilement dans ceux-ci. Ces deux objectifs à atteindre correspondent pour nous à deux modes également. Dans chaque mode, les stocks B_3 ou B_4 doivent être vidés s'ils ne le sont pas et le bras de robot doit se reconfigurer pour être dans un état adéquat à la fabrication demandée. Comme ces tâches ont une durée de réalisation relativement courte, comme peut l'être une phase d'initialisation d'une machine, nous considérons que ces deux modes sont des modes dits *transitoires*. Cependant, dans la suite de ce chapitre, ils seront conçus de la même manière que les autres modes et tel que détaillé dans le chapitre 3.

4.2.3 Logiciels utilisés

Pour traiter cet exemple, nous allons utiliser plusieurs logiciels. La premier, pour représenter les modèles et faire les premières constructions de modèles, nous utiliserons le logiciel SUPREMICA [Ake06, Ake03]. Le deuxième logiciel est DESUMA et plus particulièrement sa bibliothèque d'opérateurs sur automates appelée UMDES [Ric06]. Cette bibliothèque est codée en langage *C*, ce qui rend plus rapide les opérations faites sur des automates qui dépassent les dizaines de milliers d'états. Enfin, nous utilisons des fonctions que nous avons développées en Python [Lin05, Oli07]. Ces fonctions, en rapport avec la démarche proposée, sont la labélisation des événements et la fusion des états non significatifs.

4.3 Formalisation des exigences client

4.3.1 Constructions des modèles de composants

La première étape de la démarche est la construction des modèles des composants et la détermination des ensembles.

D'après le cahier des charges exprimé ci-dessus et la définition 22 de notre proposition, nous pouvons établir l'ensemble des composants $\mathcal{C} = \{C_1, C_2, C_3, R, L_1, L_2, P, AM, PR\}$.

Concernant les modèles des composants, ils sont définis par la définition 23 (p. 54) et sont illustrés sur la figure 4.2.

Les convoyeurs C_1 et C_2 ont un fonctionnement identique, à des labels d'événements près. Leur comportement est représenté sur la figure 4.2.(a) où l'indice i correspond au numéro de convoyeur. Les arrivées de pièces sur les convoyeurs C_1 et C_2 sont respectivement représentées par les événements s_1 et s_2. Les convoyeurs déplacent les pièces jusqu'aux stocks B_1 ou B_2 où l'arrivée des pièces dans les stocks est modélisée respectivement par les événements c_1b_1 pour le convoyeur C_1 et c_2b_2 pour le convoyeur B_2. Les pièces dans ces stocks peuvent alors être manipulées par le bras de robot R.

Remarque 7

Comme les événements représentent implicitement le flux de pièces dans le système, nous avons composé le nom des événements par le nom des machines qu'ils relient. Ainsi, pour une pièce quittant le convoyeur C_1 pour remplir le stock B_1, l'événement est nommé c_1b_1. De plus, afin de faciliter la lecture, les labels des machines ou stocks sont complètement écrit en majuscule. Les labels des états, sur les modèles, ont la première lettre en majuscule et le reste en minuscule. Enfin, les labels d'événement sont écris uniquement en minuscule. ♦

Le convoyeur C_3 a un comportement différent des autres convoyeurs. Ceci est lié au double

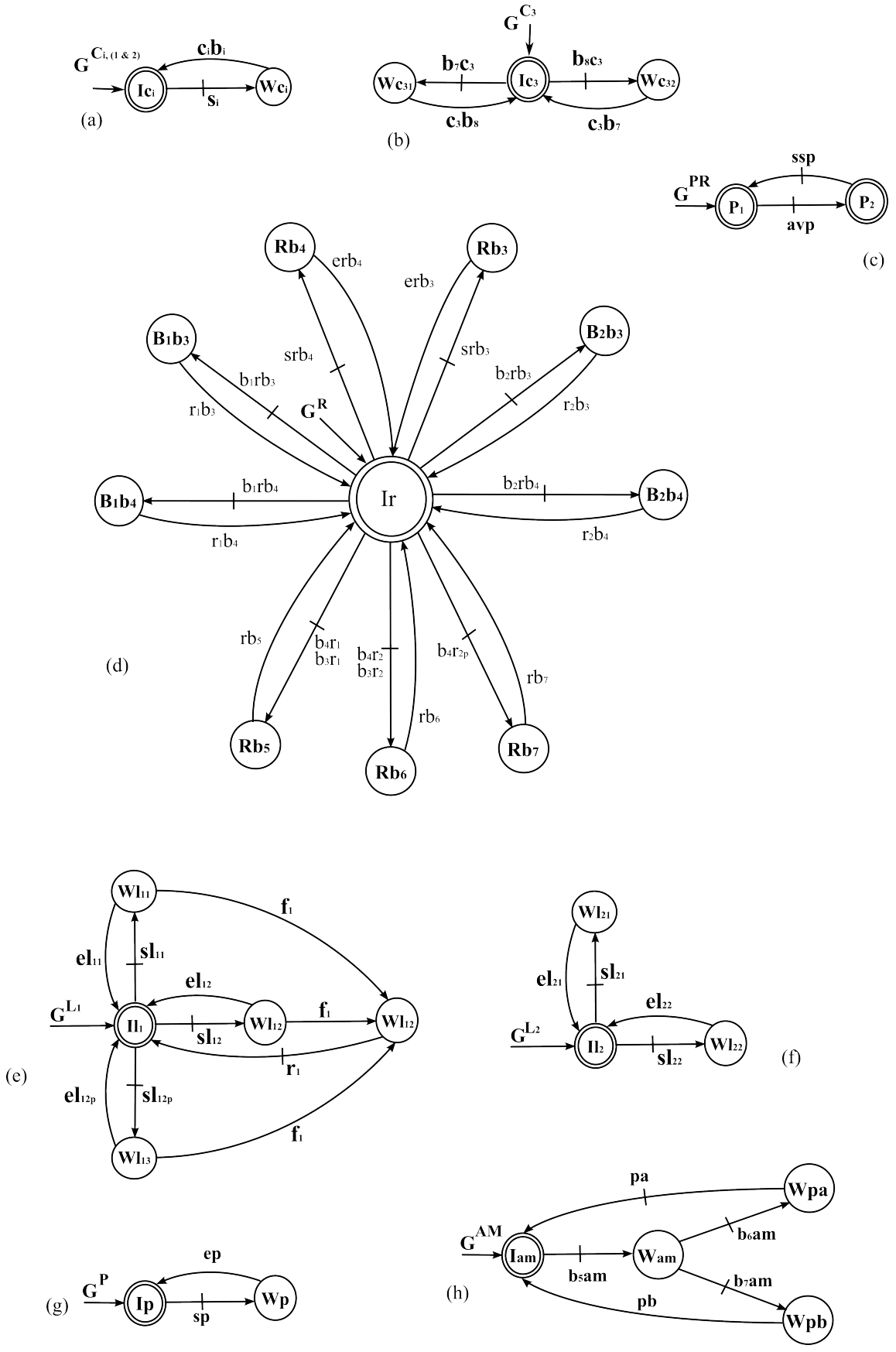

FIGURE 4.2 – Modèles des composants constituant le système flexible étudié.

sens que peuvent prendre les pièces. Le modèle de ce convoyeur est illustré par la figure 4.2.(b). Ainsi, lorsqu'une pièce est dans le stock B_7 pour être peinte, celle-ci est envoyée au convoyeur C_3 qui la déplace jusqu'au stock B_8 connecté à la machine de peinture P. Une fois peinte, la pièce est de retour dans le stock B_8 et est envoyée à nouveau vers le convoyeur C_3 pour qu'il la replace dans le stock B_7. celle-ci est maintenant prête pour être utilisée par la machine d'assemblage AM.

Le comportement du robot R est assez simplement représenté, sur la figure 4.2.(d). Le bras de robot réalise 7 actions de déplacement de pièces de stock à stock, et 2 actions de reconfiguration. Techniquement, en analysant le modèle du robot, la restriction liée à l'utilisation des stocks B_3 et B_4 simultanément vient d'une contrainte physique de sécurité pour éviter d'endommager le bras de robot., Cette restriction sera représentée par un modèle de spécification exclusivement lié au bras de robot.

Les modèles suivant, illustrés sur les figures 4.2.(e) et 4.2.(f) représentent respectivement les comportements des tours L_1 et L_2. Le comportement de la machine L_1 est plus complexe car, comme décrit dans le cahier des charges, cette machine traite une opération de plus (l'opération pour fournir des chevilles prêtes à peindre pour concevoir des goupilles coniques), mais peut également tomber en panne. Cette panne est représentée dans le modèle par l'événement f_1. Concernant la réparation, elle est représentée par l'événement r_1. Alors que la panne est logiquement un événement incontrôlable, l'événement de réparation est lui contrôlable.

Le modèle illustré sur la figure 4.2.(g) représente le comportement de la machine de peinture. Lorsqu'une pièce est chargée dans le stock B_8 pour être peinte, l'événement sp lance la tâche. Lorsque la pièce est peinte, la fin de tâche est symbolisée par l'événement ep et charge la pièce dans le stock B_8.

Enfin, la machine d'assemblage AM est représentée par le modèle de la figure 4.2.(h). Ce modèle est repris des publications citées en début de chapitre. Nous ne l'avons pas modifié bien qu'il comporte déjà certains comportements implicites. Le premier est de toujours prendre une pièce dans le stock B_5 avant de prendre la deuxième partie de pièce nécessaire à l'assemblage. Ainsi, charger le stock B_6 ou B_7 en premier ne lancera pas la tâche d'assemblage. Ensuite, la production d'une goupille conique peinte, symbolisée par l'événement pb est un événement incontrôlable alors que la production d'une goupille cylindre, symbolisée par l'événement pa, est elle contrôlable. Nous n'avons pas trouvé de justification à cette différence, hormis de provoquer un blocage possible du système dans certains cas spécifiques [1].

Nous avons omis de parler du modèle illustré par la figure 4.2.(c). Il ne s'agit pas à proprement parlé d'un composant physique du système, tel qu'on les voit sur la figure 4.1, mais du comportement de l'interrupteur que peut actionner le technicien pour activer ou désactiver la fonction de peinture et de changement de production du type de goupille. À l'état initial, nous

1. Cependant, nous n'avons pas remarqué de comportement dangereux dans nos modèles dans la suite de ce travail.

considérons que le système produit des goupilles cylindriques. La fonction de peinture n'est pas activée. L'activation de la peinture et du changement de production sont représentés par l'événement contrôlable *avp*. De même, la désactivation de la peinture et le changement de production sont représentés par l'événement contrôlable *ssp*. Avec un tel modèle, nous ne pouvons avoir qu'une alternance des événements *avp* et *ssp*, ce qui est cohérent vis-à-vis du cahier des charges.

La figure 4.3, qui illustre le système flexible incluant les noms des événements, sert à faciliter la compréhension des modèles de spécifications qui seront manipulés dans les sections suivantes.

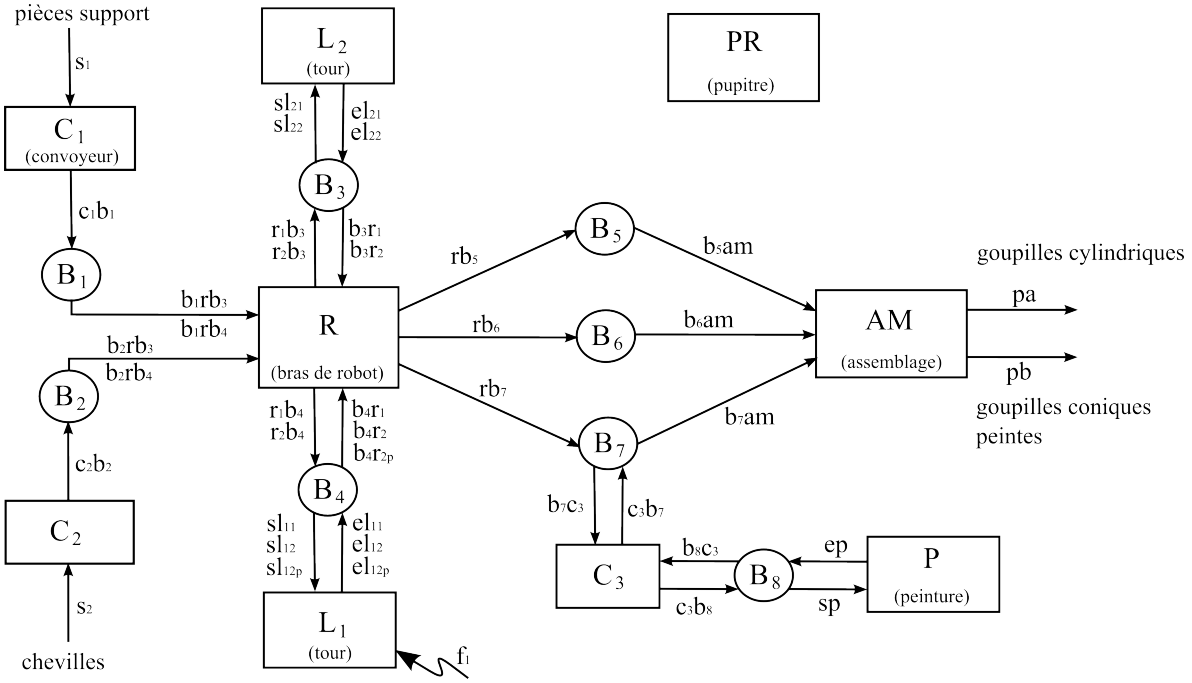

FIGURE 4.3 – FSM étudié incluant les noms des événements.

4.3.2 Construction de l'automate de modes et des ensembles de modes

Nous retenons du cahier des charges que le système devait comporter 3 modes : deux modes nominaux, que nous nommons N_1 et N_2 et qui dépendent du type de production, et un mode dégradé, nommé D, lorsque la machine L_1 tombe en panne.

Cependant, certaines tâches particulières sont à réaliser lors de commutation des modes nominaux au mode dégradé, ou inversement. Il s'agit entre autres de la reconfiguration du bras de robot R pour qu'il puisse utiliser un stock différent (le stock B_4 est utilisé pour les deux modes nominaux et le stock B_3 est utilisé pour le mode dégradé). C'est la raison pour laquelle nous faisons le choix, en tant que concepteur de la loi de commande de ce système, de créer deux modes supplémentaires. Nous les considérons comme des *modes transitoires* car il ne s'agit pas

à proprement parler de modes de fonctionnement tels qu'ils réalisent un objectif de production, mais leur but est de mettre le système dans un état adéquat selon le mode atteint par le système lors de la commutation. De plus, avoir des modes transitoires pour basculer d'un mode à l'autre permet de rajouter des contraintes de commutations restrictives sans pour autant restreindre directement le comportement d'un mode de fonctionnement. Les deux modes transitoires sont nommés par la suite Mt_1 et Mt_2.

Ainsi, nous déterminons l'ensemble de modes $\mathcal{M}$ tel que : $\mathcal{M} = \{N_1, N_2, D, Mt_1, Mt_2\}$.

Avec ces ensembles de composants et de modes, nous sommes en mesure de définir les ensembles des composants des modes ainsi que l'automate de modes. Ce dernier est illustré par la figure 4.4.

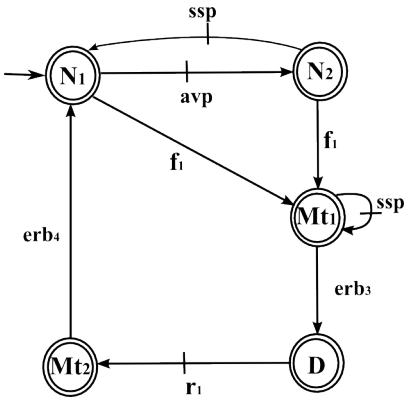

FIGURE 4.4 – Modèle de l'automate de modes $G^{\mathcal{M}}$.

Sur cette figure, les deux modes nominaux sont N_1 pour la production de goupilles cylindriques, et N_2 pour la production de goupilles coniques peintes. Le passage d'un mode à l'autre se fait par les événements générés par les requêtes venant du pupitre, dont le modèle est illustré par la figure 4.2.(c). L'occurrence d'une panne sur la machine L_1, symbolisée par l'événement f_1, fait basculer le système dans un mode transitoire Mt_1 où le bras de robot doit changer de programme d'exécution. La fin de changement de programme, symbolisée par l'événement erb_3, est l'événement déclencheur de commutation vers le mode dégradé D. Dans ce mode, le système utilise la machine L_2 et ne produit que des goupilles cylindrique. Enfin, lorsque la réparation est effective, représentée par l'événement r_1, le système commute vers un deuxième mode transitoire Mt_2 où le bras de robot change à nouveau de programme d'exécution. Là encore, la fin de ce changement, symbolisée par l'événement erb_4, est l'événement finalisant la commutation vers le mode nominal N_1.

Quand aux ensembles $\mathcal{C}^{M_j}$ représentant les composants d'un mode M_j, nous proposons le tableau suivant en guise de décomposition de ces ensembles en accord avec la définition 25.

Mode M_j	$\mathcal{C}^{M_j}$	$\mathcal{C}^{M_j}_{\circlearrowleft}$	$\mathcal{C}^{M_j}_{\leftarrow}$	$\mathcal{C}^{M_j}_{\rightarrow}$
N_1	$\{C_1, C_2, R, L_1, AM, PR\}$,	$\{C_1, C_2, R, L_1, AM\}$	$\{R, PR\}$	$\{L_1, PR\}$
N_2	$\{C_1, C_2, C_3, R, L_1, P, AM, PR\}$	$\{C_1, C_2, C_3, R, L_1, P, AM\}$	$\{PR\}$	$\{L_1, PR\}$
D	$\{C_1, C_2, R, L_1, L_2, AM\}$	$\{C_1, C_2, R, L_2, AM\}$	$\{R\}$	$\{L_1\}$
Mt_1	$\{C_1, C_2, C_3, R, L_1, P, AM\}$	$\{C_1, C_2, C_3, R, L_1, P, AM\}$	$\{L_1\}$	$\{R\}$
Mt_2	$\{C_1, C_2, R, L_1, L_2, AM\}$	$\{C_1, C_2, R, L_2, AM\}$	$\{L_1\}$	$\{R\}$

TABLE 4.1 – Ensembles $\mathcal{C}^{M_j}$ des composants suivant le mode considéré

4.3.3 Validation de la cohérence entre modèles

Cette section s'intéresse à la cohérence des modèles construits par rapport à la définition 27 (p. 57). Dans cette validation, nous sommes amenés à vérifier certaines conditions sur les modèles.

La première condition s'assure que tous les modes sont accessibles ou atteignables. Cette condition est respectée car, comme affiché par le tableau 4.4.1, aucun ensemble de mode n'est vide tel que
$\mathcal{C}^{M_j}_{\circlearrowleft} \neq \emptyset \wedge \mathcal{C}^{M_j}_{\rightleftarrows} \neq \emptyset$.

La deuxième condition s'intéresse aux ensembles $\mathcal{C}^{M_j}_{\rightarrow}$ et s'assure que les composants inclus dans ces ensembles génèrent bien un événement de commutation, défini également dans la fonction de transition de $G^{\mathcal{M}}$. Ainsi, les composants L_1, PR appartiennent bien aux modes N_1 et N_2. Cela est cohérent car le composant L_1 génère l'événement de panne f_1 faisant sortir le système de ces deux derniers modes pour atteindre le mode Mt_1. Quant au composant PR, il représente justement la commutation entre les modes N_1-N_2. Le composant L_1 concerne également le mode D, ce qui est correct car la commutation se produit sur l'événement de réparation r_1. Enfin, les modes transitoires Mt_1 et Mt_2 ont le composant R dans l'ensemble des composants de commutations sortantes. Ceci est encore une fois correct car la sortie des modes transitoires se fait sur l'occurrence de fin de reconfiguration, qui correspond aux événements erb_3 et erb_4. La condition 2 est donc respectée.

La troisième condition est équivalente à la deuxième, excepté qu'elle s'intéresse cette fois aux ensembles $\mathcal{C}^{M_j}_{\leftarrow}$. De manière analogue, il est vérifié que les composants de ces ensembles génèrent bien un événements de commutation tels qu'ils apparaissent dans la fonction de transition de l'automate de modes $G^{\mathcal{M}}$.

La condition 4 s'assure que tous les composants générant un événement impliquant une commutation soient compris dans *au moins* un ensemble $\mathcal{C}^{M_j}_{\rightleftarrows}$. Comme décrit précédemment, les événements impliquant une commutation sont les événements $avp, ssp, f_1, r_1, erb_3, erb_4$. Les événements avp et ssp sont générés par le composant PR qui est inclus dans les ensembles $\mathcal{C}^{N_1}_{\rightleftarrows}$ et $\mathcal{C}^{N_2}_{\rightleftarrows}$, les événements f_1 et r_1 sont générés par le composant L_1 qui est inclus dans tous les ensembles $\mathcal{C}^{M_j}_{\rightleftarrows}$, et enfin les événements erb_3 et erb_4 sont générés par le bras de robot R qui est inclus dans les ensembles $\mathcal{C}^{M_j}_{\rightleftarrows}$ du mode nominal N_1 et des deux modes transitoires Mt_1 et

Mt_2. Tous les composants générant un événement qui implique une commutation sont bien tous compris dans au moins un ensemble $\mathcal{C}_{\rightleftarrows}^{M_j}$. La condition est respectée.

La dernière condition vérifie que tous les événements impliquant une commutation sont compris dans l'alphabet de l'automate de modes. Ce qui est effectivement le cas.

Nous sommes alors en mesure d'affirmer que nos modèles sont cohérents entre eux. Qu'ils ne comportent aucune erreur grossière de conception, et qu'il est possible de les utiliser.

4.4 Étude intramodale

L'étude intramodale est la deuxième étape de la démarche proposée, illustrée par la figure 3.3, page 53. Cette étape consiste à étudier le comportement interne de chaque mode sans prendre en compte ni étudier le comportement commutatif. Lors de cette étape, nous construisons les modèles des procédés, les modèles des spécifications et enfin construisons les procédés sous contrôle.

4.4.1 Construction des procédés

Pour chacun des modes contenus dans $\mathcal{M}$, nous construisons les procédés $G_{in}^{M_j}$ à partir des modèles de composants G_i^C. Les composants pris pour la construction des procédés sont les composants inclus dans $\mathcal{C}_{\circlearrowleft}^{M_j}$ (tab.).

Nous donnons le tableau suivant pour indiquer l'opération de construction des modèles du procédé et la taille de ceux-ci.

$G_{in}^{M_j} = \|_{C_i \in \mathcal{C}_{\circlearrowleft}^{M_j}} G^{C_i}$	Nb. d'états	Nb. de transitions
$G_{in}^{N_1} = G^{C_1}\|\|G^{C_2}\|\|G^R\|\|G^{L_1}\|\|G^{AM}$	800	5 800
$G_{in}^{N_2} = G^{C_1}\|\|G^{C_2}\|\|G^{C_3}\|\|G^R\|\|G^{L_1}\|\|G^P\|\|G^{AM}$	4 800	46 000
$G_{in}^{D} = G^{C_1}\|\|G^{C_2}\|\|G^R\|\|G^{L_2}\|\|G^{AM}$	480	3 160
$G_{in}^{Mt_1} = G^{C_1}\|\|G^{C_2}\|\|G^{C_3}\|\|G^R\|\|G^{L_1}\|\|G^P\|\|G^{AM}$	4 800	46 000
$G_{in}^{Mt_2} = G^{C_1}\|\|G^{C_2}\|\|G^R\|\|G^{L_2}\|\|G^{AM}$	480	3 160

TABLE 4.2 – Construction et taille des modèles des procédés $G_{in}^{M_j}$

Remarque 8

Les modèles sont grands, particulièrement pour les modèles G^{N_2} et G^{Mt_1}. Cependant, il ne faut pas oublier que la taille d'un automate évolue de manière exponentielle et qu'ainsi le procédé construit par approche centralisée serait beaucoup plus important. ♦

4.4.2 construction des spécifications

Nous nous intéressons dans cette section aux modèles des spécifications. Pour le moment, seules les spécifications représentant des contraintes de fonctionnement du système *dans* le mode considéré sont prises en comptes.

Modèles des spécifications dans le mode nominal N_1

Dans ce mode, les seules contraintes qu'il est nécessaire de modéliser sont celles liées à la gestion des stocks. Cependant, tous les stocks ne sont pas utilisés. Ainsi, pour produire des goupilles cylindriques, nous avons besoin d'utiliser les stocks B_1, B_2, B_4, B_5 et B_6. Les stocks B_7 et B_3 ne sont pas à utiliser. Donc leurs événements doivent être interdits. Enfin, il n'est pas nécessaire de représenter les contraintes du stock B_8 car les événements générés par les éléments qu'il connecte ne sont pas inclus dans l'ensemble des composants utilisés $\mathcal{C}^{N_1}$. Nous proposons, les modèles de spécifications représentés sur la figure 4.5.

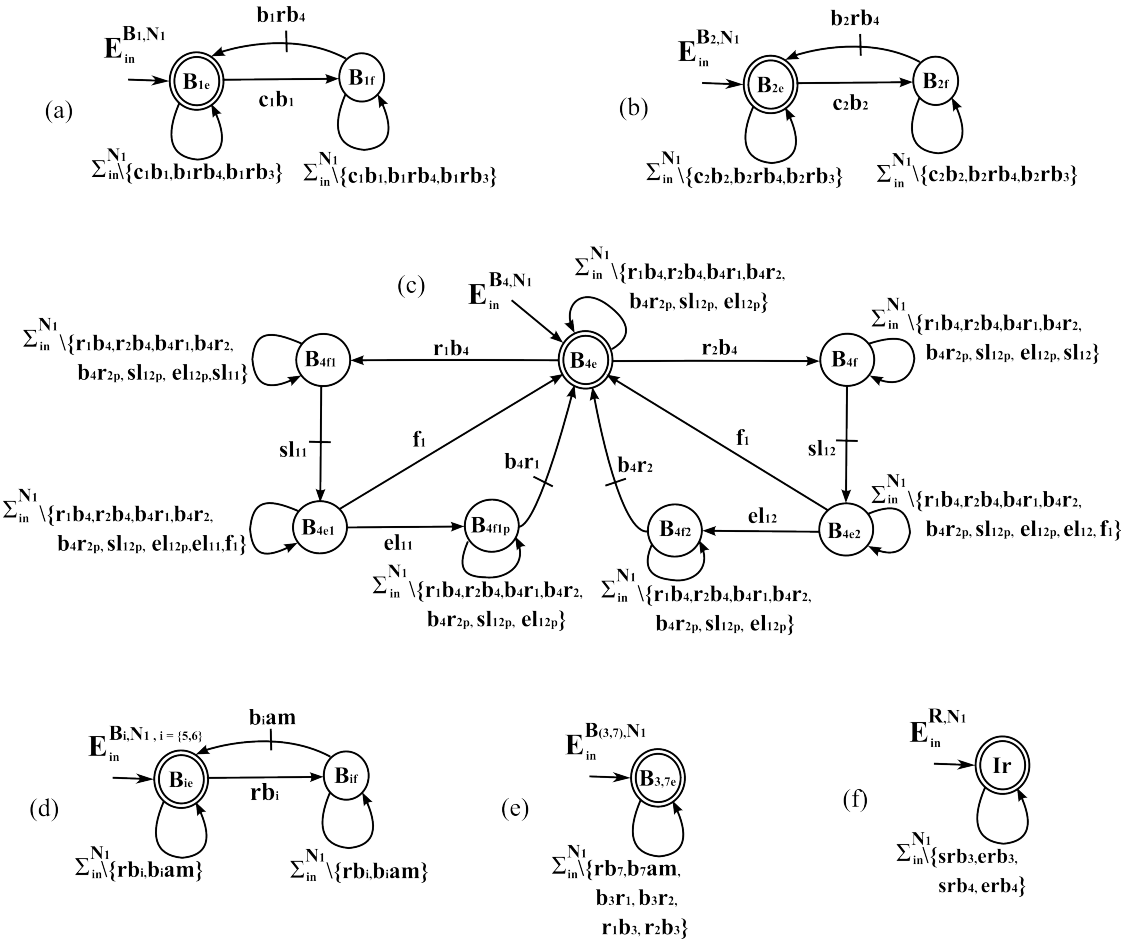

FIGURE 4.5 – Modèles des spécifications intramodales dans le mode N_1.

Les modèles des figures (a) et (b) représentent respectivement les spécifications pour les

stocks B_1 et B_2. Le modèle de la figure (c) représente la spécification du stock B_4 où seuls les événements servant à produire des goupilles cylindriques sont générés. Ainsi, les événements b_4r_{2p}, sl_{12p} et el_{12p}, représentant la production de pièces à peindre pour des goupilles coniques sont interdits. Le modèle de la figure (d) représente les contraintes pour les stocks B_5 et B_6 car, à un indice près nommé l, ils ont les mêmes contraintes à respecter. Enfin les modèles illustrés par les figures (e) et (f) représentent des événements que nous ne désirons pas et qui sont inutiles dans le comportement interne. Pour le premier modèle, il s'agit des événements des stocks B_3 et B_7. Ces événements, générés par le bras de robot, sont à interdire dans le comportement interne du mode nominal. Le deuxième représente la contrainte sur la reconfiguration du bras de robot. Comme il est inutile dans ce mode de reconfigurer le robot, nous interdisons ce comportement. Cela a pour effet de réduire la taille de l'automate final.

Modèles des spécifications dans le mode nominal N_2

Dans ce mode, nous désirons produire des goupilles coniques peintes. Pour cela, nous utilisons les composants inclus dans $\mathcal{C}_{\circlearrowleft}^{N_2}$. Concernant les modèles des spécifications, certains modèles de ce mode sont identiques aux modèles construits pour le mode nominal N_1, hormis l'ensemble des événements qui ne s'exprime plus sur Σ^{N_1} mais sur Σ^{N_2}. Ces modèles identiques sont ceux représentés par les figures 4.5(a), (b), (d) [2], (e) [3] et (f). Les modèles qui changent sont donc ceux concernant les stocks B_4, B_6 et B_7, en plus du nouveau modèle de spécification concernant B_8. Ces nouveaux modèles sont illustrés sur la figure 4.6.

La figure (a) présente la contrainte sur le stock B_4 dans le mode nominal N_2. Elle est proche de la contrainte du même stock dans le mode N_1, mais la différence se trouve sur les événements représentant l'envoi d'une pièce de type cheville. Dans le mode N_1, l'événement sl_{12} était généré après r_2b_4, alors que maintenant c'est l'événement sl_{12p}. Les figures (b) et (c) représentent respectivement les spécifications des stocks B_7 et B_8. Enfin, la figure (d) représente l'interdiction d'utiliser le stock B_6 dans ce mode.

Modèles des spécifications dans le mode dégradé D

Le mode dégradé est le comportement particulier du système quand le composant L_1 est tombé en panne. Le système utilise alors le composant de remplacement L_2 mais ne peut produire que des goupilles cylindriques. Par conséquent, ce mode n'utilise pas les composants et les contraintes liés à la peinture. Concernant les autres modèles, ceux des stocks, ceux-ci changent car les événements générés ne sont plus les mêmes. La contrainte de ne pas reconfigurer le bras de robot est également nécessaire ici, et est similaire au modèle de la figure 4.5(f) [4]. Les modèles des contraintes dans le mode dégradé D sont représentés sur la figure 4.7

2. uniquement pour le stock B_5. L'utilisation du stock B_6 est interdite
3. de même, seulement les événements du stock B_3 sont interdits, car le stock B_7 est maintenant utilisé.
4. Bien que l'ensemble des événements soit Σ^D et non Σ^{N_1}

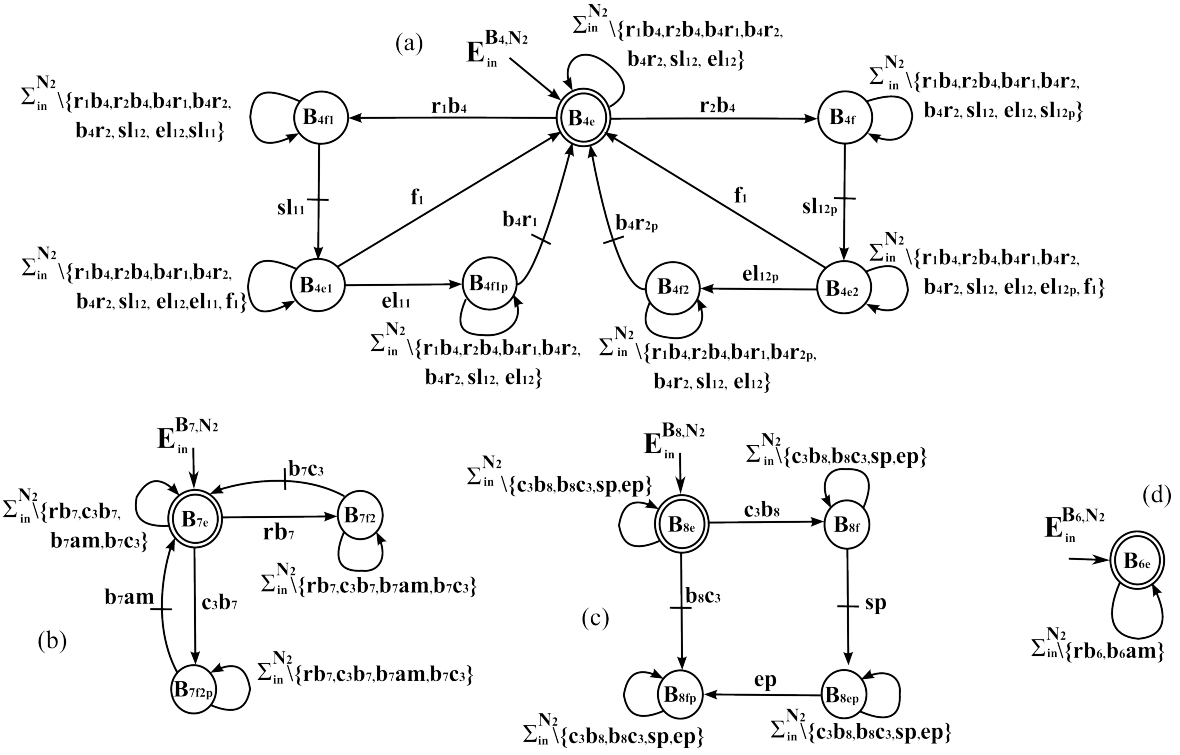

FIGURE 4.6 – Modèles des spécifications intramodales dans le mode N_2.

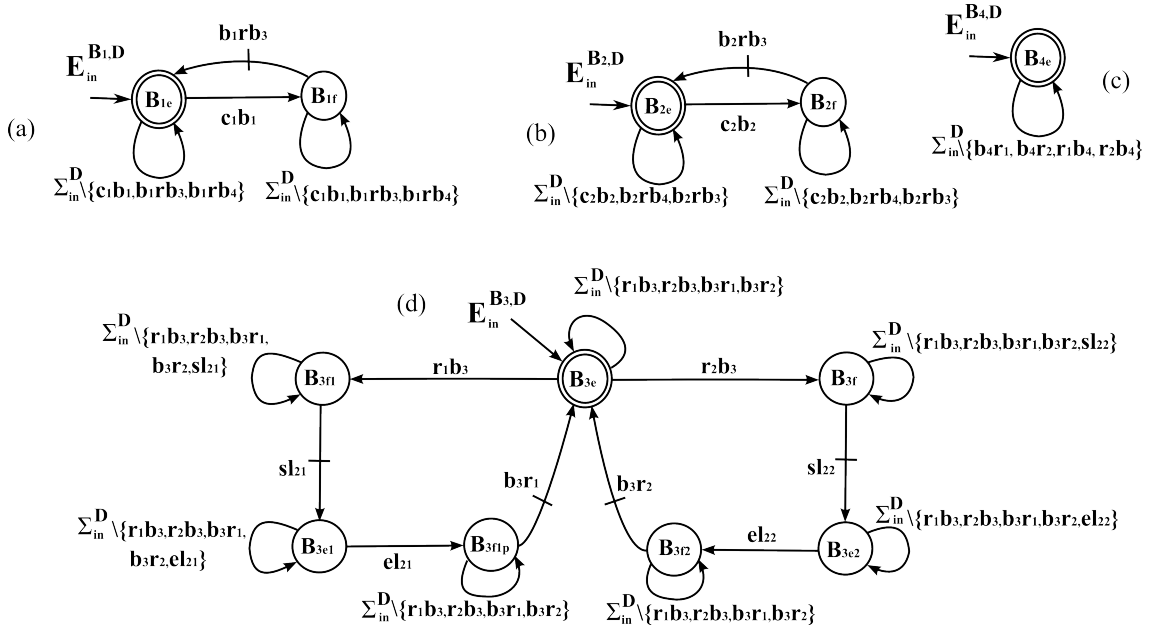

FIGURE 4.7 – Modèles des spécifications intramodales dans le mode D.

Modèles des spécifications dans les modes transitoires

Les modes transitoires sont des modes particuliers car ils ne répondent pas à un besoin lié à un objectif de production mais plutôt à des tâches très courtes à réaliser, souvent dans l'objectif de rendre le système prêt à être dans un mode de fonctionnement. Un mode transitoire est par exemple l'initialisation d'une machine. Ici, l'objectif des modes transitoires est de finir la tâche de production du mode que le système vient de quitter et de changer le programme du bras de robot afin que celui-ci puisse charger/décharger les pièces dans le stock B_3 ou B_4 suivant le mode atteint par le système. Leur but est donc de faciliter la commutation d'un mode à l'autre en séparant les spécifications de fonctionnement d'un coté et les spécifications de commutation de l'autre. Ne considérant pas encore les spécifications de commutation dans cette étude intramodale, les modèles des spécifications dans les modes transitoires sont donc celles sur la production des autres modes. Cependant, les contraintes de reconfiguration du bras de robot sont, dans ces modes, autorisées.

Ainsi, les spécifications du mode transitoire Mt_1 sont celles illustrées sur la figure 4.6, sans tenir compte des contraintes sur la reconfiguration telles qu'illustrées sur la figure 4.5(f).

Le mode transitoire Mt_2 doit respecter les mêmes contraintes que celles du mode dégradé D, sans tenir compte de celles sur la reconfiguration du bras de robot.

Taille des modèles des spécifications pour l'étude intramodale

Nous donnons ici un tableau contenant un rappel des modèles de spécifications utilisés et la taille de ces modèles.

$E_{in}^{M_j} = \times_l E^{l,M_j}$	Nb. d'états / transitions
$E_{in}^{N_1} = E_{in}^{b_1,N_1} \times E_{in}^{b_2,N_1} \times E_{in}^{R,N_1} \times E_{in}^{b_3,N_1} \times E_{in}^{b_4,N_1} \times E_{in}^{b_5,N_1} \times E_{in}^{b_6,N_1} \times E_{in}^{b_7,N_1}$	112 / 608
$E_{in}^{N_2} = E_{in}^{b_1,N_2} \times E_{in}^{b_2,N_2} \times E_{in}^{R,N_2} \times E_{in}^{b_3,N_2} \times E_{in}^{b_4,N_2} \times E_{in}^{b_5,N_2} \times E_{in}^{b_6,N_2} \times E_{in}^{b_7,N_2} \times E_{in}^{b_8,N_2}$	672 / 4 544
$E_{in}^{D} = E_{in}^{b_1,D} \times E_{in}^{b_2,D} \times E_{in}^{R,D} \times E_{in}^{b_3,D} \times E_{in}^{b_4,D} \times E_{in}^{b_5,D} \times E_{in}^{b_6,D} \times E_{in}^{b_7,D}$	112 / 576
$E_{in}^{Mt_1} = E_{in}^{b_1,Mt_1} \times E_{in}^{b_2,Mt_1} \times E_{in}^{b_3,Mt_1} \times E_{in}^{b_4,Mt_1} \times E_{in}^{b_5,Mt_1} \times E_{in}^{b_6,Mt_1} \times E_{in}^{b_7,Mt_1} \times E_{in}^{b_8,Mt_1}$	1 728 / 13 632
$E_{in}^{Mt_2} = E_{in}^{b_1,Mt_2} \times E_{in}^{b_2,Mt_2} \times E_{in}^{b_3,Mt_2} \times E_{in}^{b_4,Mt_2} \times E_{in}^{b_5,Mt_2} \times E_{in}^{b_6,Mt_2} \times E_{in}^{b_7,Mt_2}$	112 / 576

TABLE 4.3 – Construction des modèles de spécifications $E_{in}^{M_j}$

4.4.3 Construction des procédés sous contrôle interne

À partir des procédés et des modèles de spécifications dans chaque mode, nous sommes en mesure de construire les procédés sous contrôle. La construction des modèles H^{M_j} est décrite par la définition 28 (p. 61). Nous ne pouvons pas représenter les modèles, car ceux-ci ont une taille trop importante, toutefois nous proposons le tableau récapitulatif suivant, qui indique la taille de ceux-ci.

$H_{in}^{M_j}$	Nb. d'états	Nb. de transitions
$H_{in}^{N_1}$	1 458	4 725
$H_{in}^{N_2}$	9 288	41 805
H_{in}^{D}	972	2 808
$H_{in}^{Mt_1}$	42 174	215 145
$H_{in}^{Mt_2}$	2 322	9 018

TABLE 4.4 – Construction des procédés sous contrôle interne $H_{in}^{M_j}$

Remarque 9

Les modèles sont de petites tailles, excepté le modèle $H_{in}^{Mt_1}$. Ceci est dû à deux facteurs. Le premier est le nombre de composants utilisés. L'ensemble des composants utilisé dans le mode N_2 contient deux éléments (L_2 et PR) de moins que l'ensemble des composants du système. Pris ensemble, ces deux composants représentent un automate à 6 états, et augmentera alors la taille du modèle du procédé par 6. Le deuxième facteur concerne l'ensemble des spécifications représentées qui correspondent également à une grande partie des spécifications du système global bien que n'incluant pas le modèle de spécifications du stock B_3. La taille du modèle incluant toutes les spécifications, qui serait construites par approche centralisée, est alors bien plus grande que le plus grand des modèles mentionnés dans ce tableau. ♦

4.4.4 Validation de l'étude intramodale

Premièrement, les procédés sous contrôle interne $H_{in}^{M_j}$ existent. Ce qui prouve qu'il est possible de trouver une trajectoire dans chaque mode qui respectent toutes les spécifications. Deuxièmement, il faut que ces modèles soit validés par le client. Pour ce faire, l'étude de scénarios peut être réalisée. Ces scénarios peuvent porter sur des trajectoires de production importantes pour le client. C'est la première étape où il est nécessaire de faire valider les modèles par le client avant de continuer. Des erreurs non repérées à cette étape conduiraient à les laisser dans les modèles suivants. Dans cet exemple, après l'étude de scénarios possibles, nous validons les modèles créés qui ne possèdent pas d'erreur apparente. Nous pouvons passer à l'étape d'après qui concerne l'étude des commutations.

Remarque 10

Nous souhaitons faire une remarque à ce point. Non pour signaler une chose dont le lecteur devrait faire attention, mais pour signaler ce qui s'est passé lors de la réalisation de cet exemple.

En effet, nous avions fait une erreur sur un de nos modèles. Plus précisément, nous avions mal conçu le modèle du convoyeur C_3. En effet, le modèle que nous avions créé se comportait ainsi : lorsqu'une pièce était envoyée du stock B_7 au convoyeur C_3, ce dernier renvoyait immédiatement la pièce dans le stock B_7 au lieu de l'envoyer au stock B_8 pour la peindre. De même, une pièce envoyée au convoyeur depuis le stock B_8 repartait dans ce même stock. Ainsi,

une pièce ne pouvait jamais être peinte. Notre modèle $H_{in}^{N_2}$ *ne faisait que* 4.716 *états et* 20.277 *transitions. Cependant, bien que le modèle existe, l'étude du modèle par des scénarios a mis en évidence ce problème. Nous l'avons ensuite corrigé et refait les tests actuels qui se sont révélés corrects.*

En disant cela, nous souhaitons mettre en lumière certes une erreur de conception de notre part, mais également la correction de celle-ci au début de la démarche. Cette erreur, assez simple, aurait pu se répercuter dans les modèles suivants et nous aurions pu ne pas nous en apercevoir avant la livraison des modèles. En effet, aucune vérification telle que : les pièces sont-telles bien peintes ? n'existe. Or si cette vérification est réalisée, elle aurait été fait a posteriori *des derniers modèles construits.* ♦

4.5 Étude intermodale et suivi de trajectoires

L'étude intramodale se concentre seulement sur les composants utilisés pour représenter le comportement interne du système dans les modes. L'étude intermodale, troisième étape de la démarche illustrée par la figure 3.3 (p. 53), concerne l'extension des modèles pour créer toutes les commutations ainsi que la conformité de ces commutations à l'égard des spécifications. Nous incluons également dans cette section le suivi de trajectoire qui est l'étape 4 de la démarche. Ces deux étapes sont fortement liées car lorsque le suivi de trajectoire identifie des commutations comme incompatibles ou inconsistance, il faut alors une spécification supplémentaire pour les interdire. Cet ajout s'opère dans une nouvelle étude intermodale.

Ainsi, dans les prochains sections, notamment la partie sur les spécifications de modes, nous donnons toutes les spécifications nécessaires pour chaque mode, dont les spécifications complémentaires pour obtenir des modèles consistants.

4.5.1 Extension des modèles des procédés

Nous considérons maintenant dans chaque mode tous les composants, inclus dans $\mathcal{C}^{M_j}$, et non plus seulement ceux représentant le comportement interne inclus dans $\mathcal{C}_{\circlearrowleft}^{M_j}$. De plus, nous incluons également le comportement de l'automate de modes $G^{\mathcal{M}}$, illustré pr la figure 4.4, pour ajouter le comportement modal.

Le procédé G^{M_j} dans un mode est donc défini par :$G^{M_j} = G^{\mathcal{M}} |||[|||_{C_k \in \mathcal{C}^{M_j}} G^{C_k}]$. Nous proposons le tableau suivant pour l'opération de construction, ainsi que la taille des modèles des procédés.

4.5.2 Construction des modèles des spécifications intermodales

Les modèles des procédés G^{M_j} contiennent le comportement interne précédemment validé et toutes les trajectoires de commutations possibles. À cette étape il convient de représenter

G^{M_j}	Nb. d'états	Nb. de transitions
$G^{N_1} = G^{\mathcal{M}} \|\| G^{C_1} \|\| G^{C_2} \|\| G^{R} \|\| G^{L_1} \|\| G^{AM} \|\| G^{PR}$	3 840	27 552
$G^{N_2} = G^{\mathcal{M}} \|\| G^{C_1} \|\| G^{C_2} \|\| G^{C_3} \|\| G^{R} \|\| G^{L_1} \|\| G^{P} \|\| G^{AM} \|\| G^{PR}$	23 040	219 072
$G^{D} = G^{\mathcal{M}} \|\| G^{C_1} \|\| G^{C_2} \|\| G^{R} \|\| G^{L_2} \|\| G^{AM} \|\| G^{L_1}$	6 720	59 456
$G^{Mt_1} = G^{\mathcal{M}} \|\| G^{C_1} \|\| G^{C_2} \|\| G^{C_3} \|\| G^{R} \|\| G^{L_1} \|\| G^{P} \|\| G^{AM}$	13 440	132 352
$G^{Mt_2} = G^{\mathcal{M}} \|\| G^{C_1} \|\| G^{C_2} \|\| G^{R} \|\| G^{L_2} \|\| G^{AM} \|\| G^{L_1}$	6 720	59 456

TABLE 4.5 – Construction des procédés G^{M_j}

les spécifications intermodales. Ces spécifications intermodales sont composées de deux types de spécification : les spécifications intramodales étendues, notées $E_{\circlearrowleft}^{M_j}$, sont construites à partir des spécifications intramodales $E_{in}^{M_j}$ auxquelles nous rajoutons des comportements qui n'étaient pas précédemment modélisés et qui sont liés aux événements générés par les nouveaux composants. Le deuxième type de spécifications sont les spécifications de commutation, notée $E_{\rightleftarrows}^{M_j}$, qui représentent les contraintes sur les spécifications exprimées dans le cahier des charges.

Remarque 11

Par facilité, et au détriment du nombre de pages, nous représentons pour chaque mode tous les modèles de spécifications utilisés. ♦

Modèles des spécifications dans le mode nominal N_1

Les modèles des spécifications dans le mode nominal N_1 sont illustrés sur la figure 4.8.

Les figures 4.8(a) et 4.8(b) sont des modèles étendus des figures 4.5(a) et 4.5(b) qui ont été construites dans l'étude intramodale. Leur comportement est étendu en permettant l'occurrence des événements b_1rb_3 et b_2rb3 qui étaient précédemment systématiquement interdits dans le comportement interne. Il en est de même avec la figure 4.8(c) qui représente la spécification pour le stock B_4. Ce modèle prend en compte dorénavant toutes les pièces qui passent par lui et non seulement celles du mode considéré.

Les figures 4.8(d) et 4.8(e) représentent les spécifications des stocks B_5 et B_6 et ont les mêmes modèles que dans l'étude intramodale. Ceci n'est pas le cas de la figure 4.8(f) qui est également étendue. Elle concerne la spécification du stock B_7 et représente le comportement d'une pièce qui arrive dans le stock et qui en repart pour être assemblée. Ainsi, dans ce mode, ce modèle ne prend en compte qu'une partie du comportement global de B_7, cependant la partie représentée est celle commune avec le reste. Cela permet de garder une trace des informations issues de ce stock, s'il est utilisé. De manière équivalente, le modèle illustré par la figure 4.8(g) représente la spécification du stock B_3 vue par les seuls événements appartenant à cet alphabet de mode. Bien que ce stock ne soit pas utilisé dans le mode considéré N_1, il peut l'être dans un autre mode et nous devons garder une trace des événements produits afin d'être en mesure de déterminer si le stock B_3 est rempli ou non lors d'une activation du mode N_1.

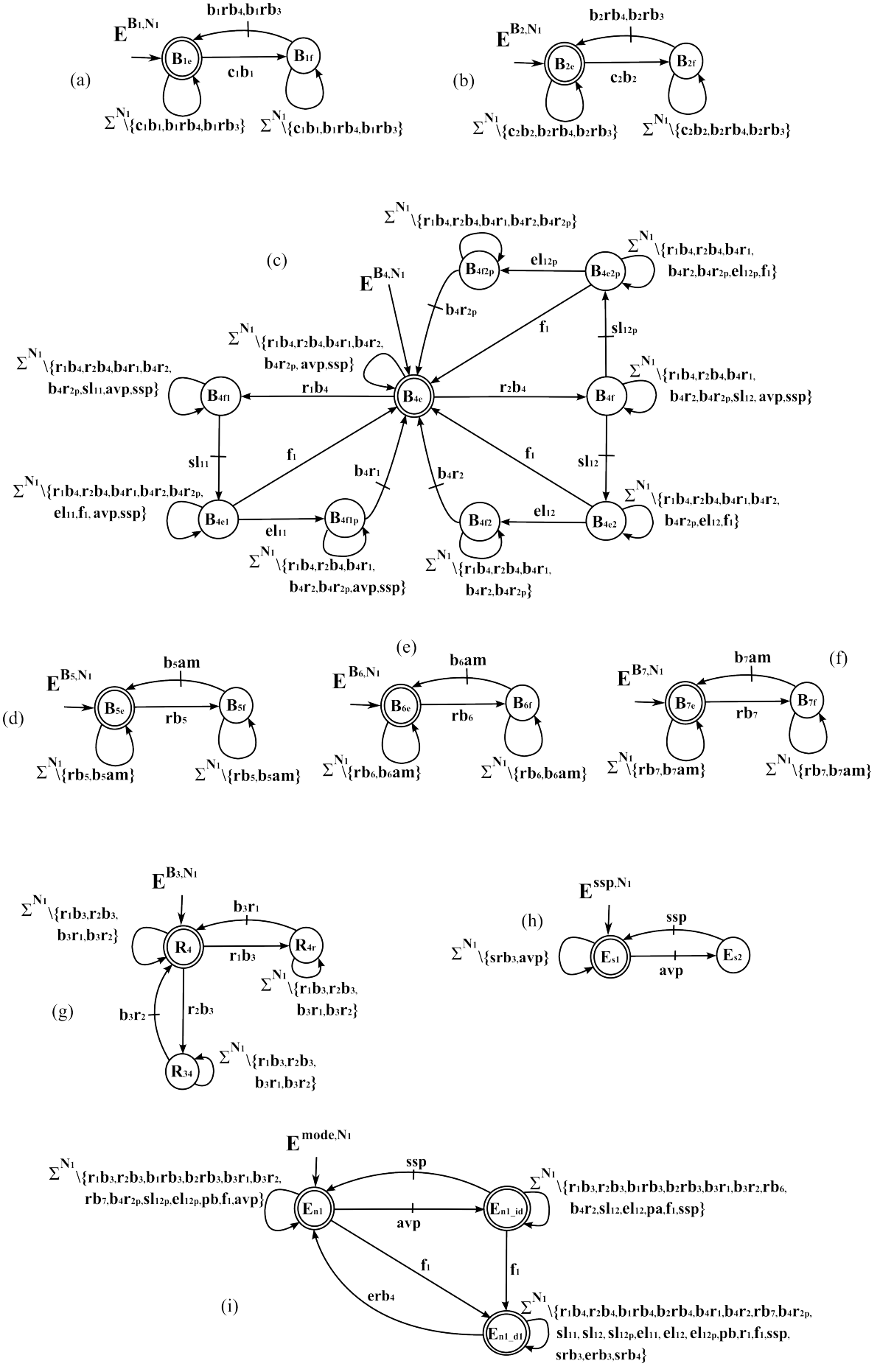

FIGURE 4.8 – Modèles des spécifications intermodales dans le mode N_1.

Deux comportements sont ajoutés à ces spécifications. Ce sont des spécifications liées exclusivement aux commutations et non à une extension des modèles pour garder une trace des informations ou pour inclure de nouveaux événements. Le premier est généré par le modèle illustré sur le figure 4.8(h). Cette spécification contraint le système à générer l'événement *ssp* s'il y a eu une panne précédemment et qu'il y a eu l'événement *avp*. La raison est la suivante. Pour quitter le mode N_1, il faut soit l'occurrence de l'événement *avp*, soit l'occurrence de l'événement f_1. Or, après l'occurrence de l'événement *avp*, le système s'attend à revenir dans N_1 dans un état où la peinture n'est pas activée, et où il peut l'activer. Cependant, si la spécification représentée par ce modèle ne forçait pas l'événement *ssp*, le système aurait pu générer les événements *avp* suivi de f_1 et revenir dans le mode N_1 sans avoir désactivé la peinture. Nous étions là face à une inconsistance. Celle-ci fut détectée par le *suivi de trajectoire* et nous a contraint à rajouter cette spécification.

Le suivi de trajectoire a également mis à jour la nécessité du deuxième comportement. Par mesure de simplicité, cette restriction a été ajoutée à la spécification du stock B_4 illustrée par la figure 4.8(c). Dans cette figure, les événements *avp* et *ssp*, c'est-à-dire les événements générant l'activation et la désactivation de la peinture et du changement de pièces à produire, ne peuvent se produire *que* si le stock est non utilisé, ou utilisé pour produire une pièce support, symbolisé par l'événement r_1b_4, ou qui contient une cheville, symbolisée par l'événement r_2b_4, qui n'a pas encore été envoyée à la machine L_1. En effet, avec les spécifications illustrées sur cette page, il est possible de générer un des deux événements de commutation pendant que la machine L_1 produisait une cheville. Si cela se produisait, le système se bloquerait le temps de revenir à l'ancien mode. Le suivi de trajectoire a détecté une incompatibilité à ce niveau qui a été corrigée par l'ajout de cette restriction.

Enfin, la dernière figure 4.8(i) représente la spécification de mode. C'est-à-dire le comportement autorisé quand le mode est actif et le comportement autorisé quand le mode est inactif. Cependant, ici, deux états ne sont pas suffisants. Ceci est lié à l'explication donnée concernant la figure 4.8(h). Il est nécessaire de faire une distinction sur la façon dont le mode est quitté, et surtout de garder un minimum d'informations sur ce qui se passe une fois celui-ci désactivé. Sinon, le suivi de trajectoire aurait détecté un problème de consistance entre les modèles.

Modèles des spécifications dans le mode nominal N_2

Les modèles des spécifications dans le mode nominal N_2 sont illustrés sur la figure 4.9.

Dans ce mode, les spécifications liées aux stocks B_1, B_2, B_3, B_4, B_5, et B_6, représentées par les modèles des figures 4.9(a),(b),(c),(e),(f) et (j), sont identiques [5] à ceux considérés dans les spécifications intermodales pour le mode N_1 illustrées figure 4.8.

La spécification concernant le stock B_8, illustrée sur la figure 4.9(d), est une spécification

5. à un alphabet près

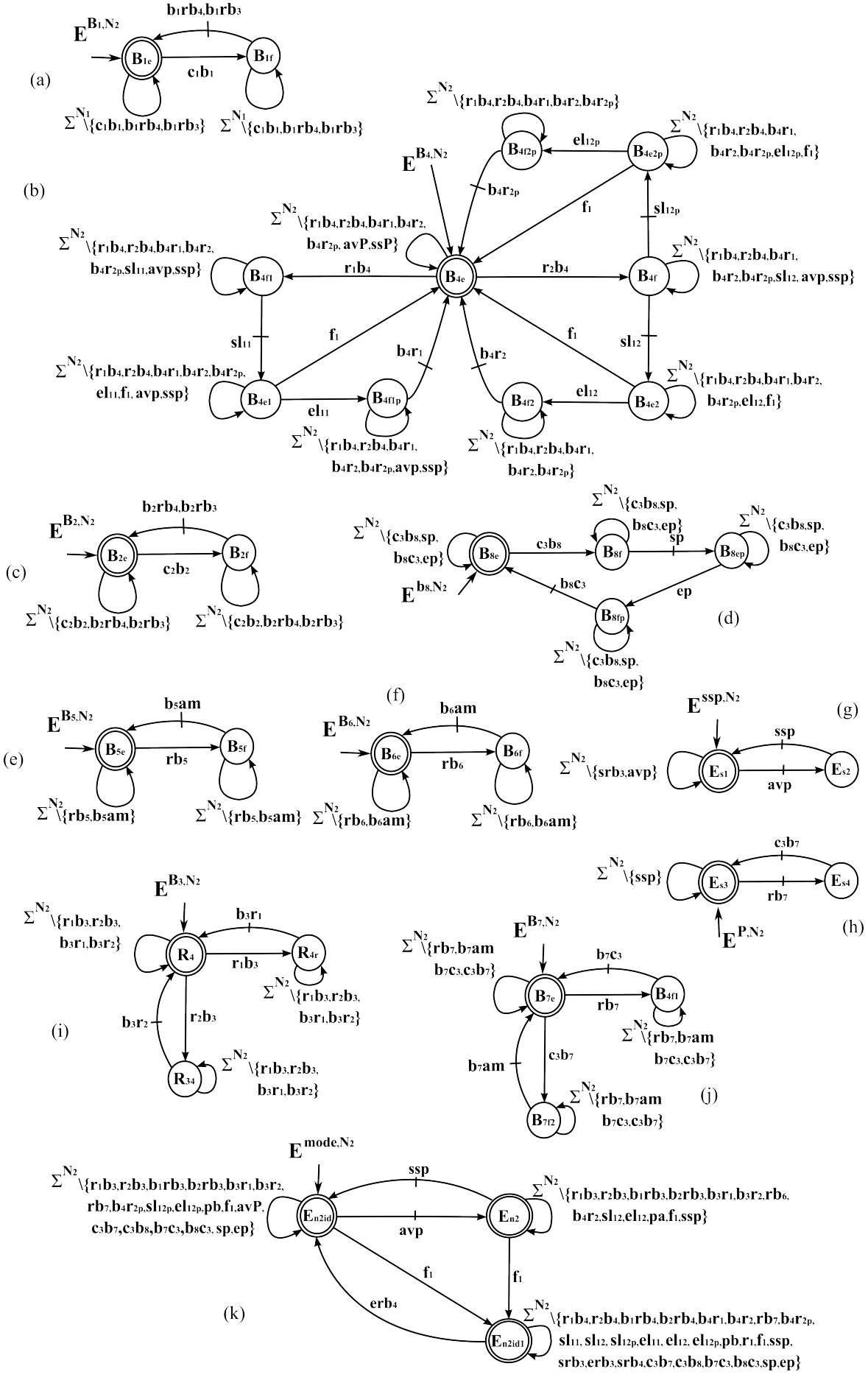

FIGURE 4.9 – Modèles des spécifications intermodales dans le mode N_2.

intramodale. Il en va de même pour la spécification du stock B_7 illustrée sur la figure 4.9(i).

Les spécifications nouvelles, liées à la commutation, sont celles représentées par les figures 4.9(g), 4.9(h) et 4.9(k). Les figures 4.9(g) et 4.9(k) sont des modèles équivalents sont utilisés dans les spécifications du mode N_1. La première concerne la nécessité de désactivé *ssp* si un événement de panne a eu lieu dans N_2 avant de basculer dans N_1 et la deuxième est la spécification de mode. Cependant l'ensemble d'événements sur les états est différent.

La figure 4.9(h) est en revanche une nouvelle spécification. Elle est rendue nécessaire pour éviter une inconsistance mise en évidence par le suivi de trajectoire. En effet, si un événement de désactivation de peinture, symbolisé par *ssp*, se produit alors qu'une pièce se trouve entre le stock B_7 et la machine de peinture P, elle ne sera pas vue lors du changement de mode. Il y aura une perte d'information liée au fait que plusieurs tous les événements des machines C_3 et P projetés sur l'alphabet du mode N_1 seront effacés comme s'ils ne s'étaient jamais produit. Ainsi, une pièce peut se trouver sur une de ces deux machines ou dans le stock B_8 alors que le système considérera après un changement de mode qu'aucune pièce n'y est. Il est donc nécessaire de rajouter une spécification qui contraint à ne pas désactiver la peinture si une pièce est en cours de traitement. Dis autrement, si une pièce arrive dans le stock B_7, tant que cette pièce n'est pas peinte nous n'autorisons pas la désactivation de la peinture. Ainsi, si le stock B_7 est plein, l'information non perdue dans les autres modes est que le stock est plein et qu'une pièce déjà peinte attend d'être utilisée pour assembler une goupille conique. Nous sommes alors en mesure de savoir précisément l'état du système quand il activera N_2.

Modèles des spécifications dans le mode transitoire Mt_1

Le mode transitoire Mt_1 est fortement lié aux comportement des modes nominaux. En effet, nous l'avons créé pour simplifier les commutations des modes nominaux vers le mode dégradé. De plus, certaines spécifications n'ont besoin d'être respectées que durant un fonctionnement très court. Ainsi, les modèles des spécifications dans le mode transitoire Mt_1 sont les mêmes que celles du mode N_2, illustrées sur la figure 4.9, auxquelles nous ajoutons deux spécifications supplémentaires illustrées sur la figure 4.10.

La première spécification modélisée sur la figure 4.10 est équivalente à trois spécifications déjà exprimées dans d'autres modes. Ce sont celles représentées par les figure 4.8(h), 4.9(g) et 4.9(h) qui concernent la fin de la tâche peinture avant de commuter et l'obligation de désactiver la peinture après la survenue d'une panne.

La deuxième spécification est celle concernant la reconfiguration du bras de robot, mentionnée dans le cahier des charges. Jusqu'à maintenant, le comportement de reconfiguration n'était pas étudié dans les modes nominaux. En effet, en dehors de l'ordre des événements générés, il n'est pas nécessaire d'être pris en compte systématiquement, seulement dans les modes transitoires comme c'est le cas ici. Dans cette spécification, deux comportements sont décrits. Le premier ne concerne que les événements générés par le robot. Dans l'état initial, le bras de ro-

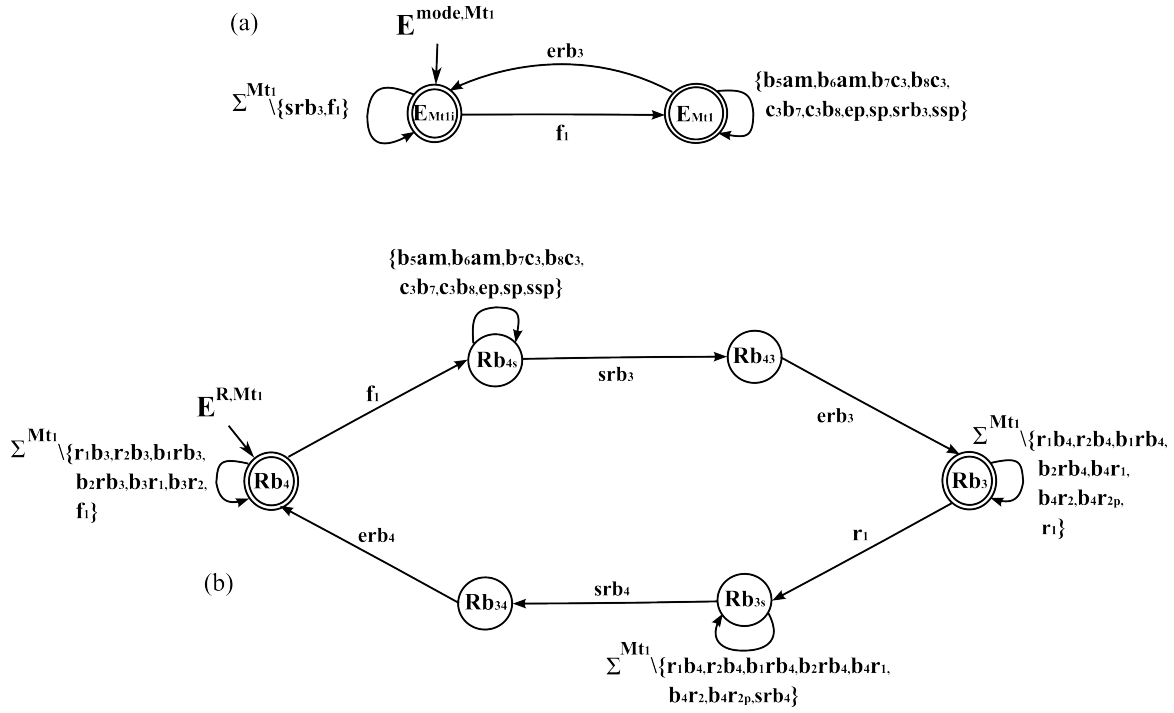

FIGURE 4.10 – Modèles des spécifications intermodales dans le mode transitoire Mt_1.

bot ne peut utiliser que le stock B_4, alors qu'une fois reconfiguré, il ne peut utiliser que le stock B_3. Cela décrit bien le cahier des charges. Le deuxième comportement est l'interdiction de tout événement une fois une reconfiguration lancée (événements srb_3 et srb_4) jusqu'à la fin de reconfiguration symbolisée par erb_3 et erb_4. Cela permet d'éviter la génération supplémentaire d'événement *juste* avant la commutation dans un nouveau mode, car il faut rappeler que ces derniers événements sont ceux qui font commuter le système.

Modèles des spécifications dans le mode dégradé D

Les modèles de spécification du mode dégradé D sont illustrés sur la figure 4.11.

Les spécifications sur les stocks B_1, B_2, B_5 et B_6 sont équivalentes à celles vues dans les modes précédents. La spécification liée au stock B_4 en est proche, car elle ne restreint plus les occurrences des événements avp et ssp. En effet, le mode dégradé ne s'intéresse pas à des trajectoires précises. Elles lui sont communiquées par le suivi de trajectoire. Il peut ainsi générer plus de comportements que nécessaire. Cela permet de simplifier la création des modèles de spécification. En revanche, le risque est d'augmenter la taille des modèles. Le modèle sur le stock B_7, représenté sur la figure 4.11(f) est équivalent à celui utilisé dans le mode nominal N_1. En effet, à partir de l'ensemble des composants utilisés, l'alphabet du mode dégradé ne contient pas tous les événements du stock B_7, notamment ceux de la partie peinture.

La spécification illustrée sur la figure 4.11(g) concerne le stock B_3. C'est une spécification intramodale déjà créée et représentée sur la figure 4.7(d). Cependant, nous avons rajouté dans

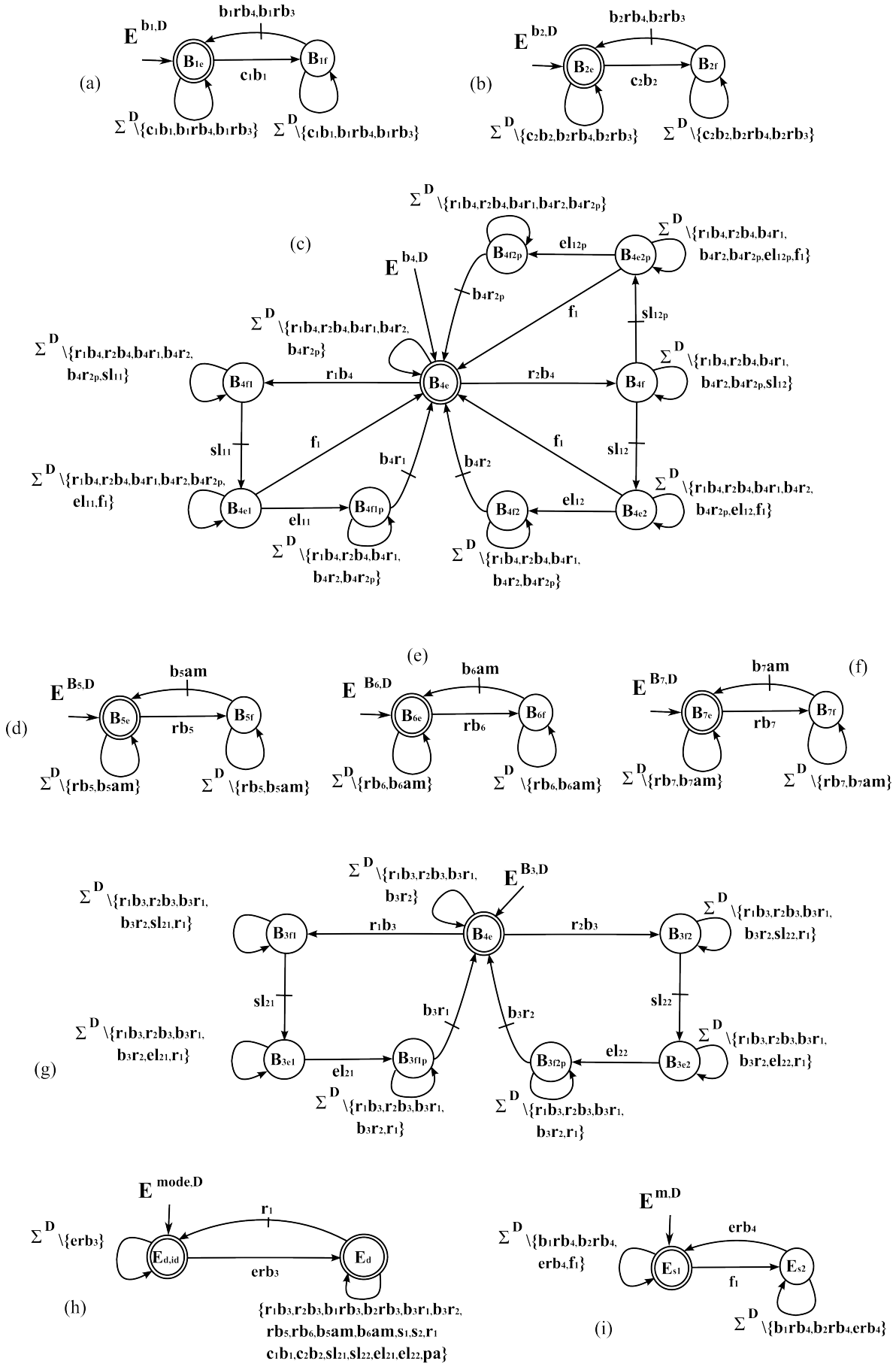

FIGURE 4.11 – Modèles des spécifications intermodales dans le mode *D*.

cette spécification une contrainte supplémentaire. Nous n'autorisons un événement de réparation que lorsque le stock B_3 n'est pas utilisé. En effet, le cahier des charges demande à ce que le stock B_3 soit vide pour commuter. Cependant, lorsqu'une pièce est envoyé à la machine L_2, le stock B_3 est considéré comme vide. Il pouvait alors se produire un événement de commutation r_1 qui conduisait à une inconsistance.

La spécification illustrée sur la figure 4.11(h) est la spécification de mode. Elle autorise tous les événements quand ce mode est désactivé et elle n'autorise que ceux utiles au mode dégradé lorsque ce dernier s'active. Ici, un autre avantage à séparer les spécifications de mode vis-à-vis d'un modèle monolithique est que ce modèle, dans ce mode, est très simple à concevoir. C'est parce que la fonction de suivi de trajectoire a détecté une inconsistance que la spécification illustrée sur la figure 4.11(i) est créée. En l'état, cette spécification représente : si un événement de panne survient, alors on interdit au système d'envoyer des pièces vers le stock B_4 tant que la commutation vers le mode dégradé n'est pas effective. En effet, en laissant un comportement très permissif dans la spécification de mode, quand le mode est désactivé, le suivi de trajectoire a identifié des traces où après une panne, le bras de robot remplissait le stock B_4 car ce dernier était vide. Évidemment, il ne le vidait pas ensuite. Néanmoins, cet état "rempli" du stock B_4 en activant le mode dégradé D est à la fois contraire au cahier des charges, mais génère également des trajectoires de commutation menant à r_1 qui n'existe pas dans le mode transitoire Mt_2. Ceci est due à des spécifications différentes dans le mode transitoire Mt_2 qui interdisaient de fait ces trajectoires.

Modèles des spécifications dans le mode transitoire Mt_2

Les modèles des spécifications dans le mode transitoire Mt_2 sont en partie identiques à ceux illustrés sur la figure 4.11. Seuls ceux qui changent ou sont en suppléments sont illustrés sur la figure 4.12.

Les modèles de spécifications dans ce mode sont similaires à ceux du mode dégradé. Toutefois, comme il s'agit d'un mode transitoire, servant à faciliter le changement de mode entre le dégradé et le nominal, il contient un ensemble d'événements plus restrictif. En effet, il n'est pas intéressant dans ce mode de lancer la production d'une pièce. Tout ce qui importe est d'être dans un état cohérent pour commuter vers le mode nominal, et que le bras de robot puisse se reconfigurer pour utiliser le stock B_4. Ainsi, la spécification de mode, représentée sur la figure 4.12(b) a un comportement plus restrictif lorsque ce mode est activé. La modification apportée au modèle de la spécification du stock B_3, illustrée figure 4.12(a), concerne l'autorisation de générer un événement erb_4 uniquement lorsque le stock n'est pas utilisé. Ce rajout est équivalent au modèle de la figure 4.11(i) qui disparaît ici. Enfin, la figure 4.12(c) illustre la spécification de reconfiguration du bras de robot. C'est la même que celle utilisée dans les spécifications du mode transitoire Mt_1 hormis que l'alphabet sur lequel il s'exprime est différent.

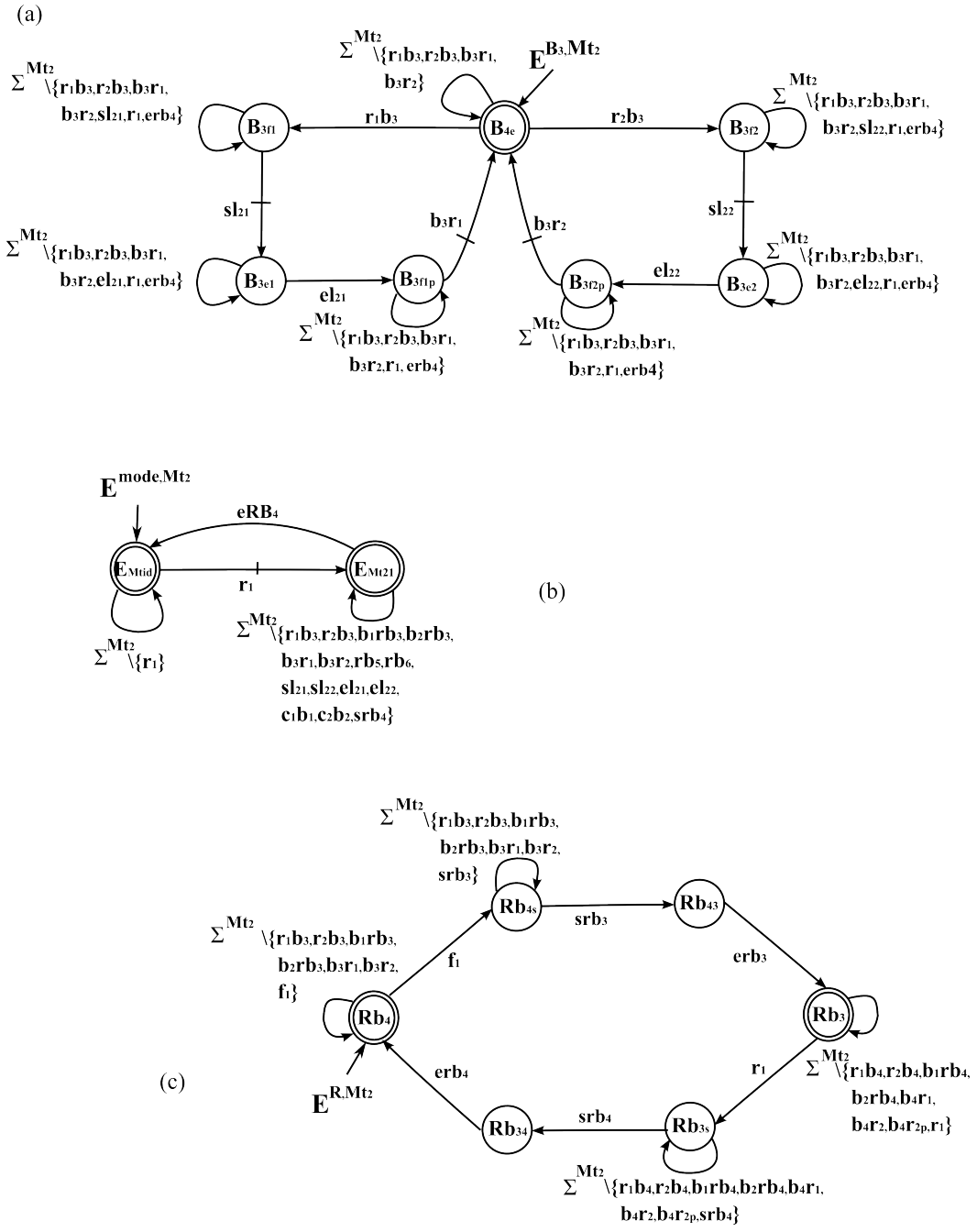

FIGURE 4.12 – Modèles des spécifications intermodales dans le mode transitoire Mt_2.

Taille des modèles des spécifications pour l'étude intramodale

Nous avons détaillé pour chacun des modes les spécifications considérées, et leur modèle. En prenant en compte toutes ces spécifications, c'est-à-dire les spécifications intramodales étendues, les spécifications de commutations et les spécifications rajoutées suite à une inconsistance identifiée dans le suivi de trajectoire, nous sommes en mesure d'évaluer la taille du modèle des spécifications dans chaque mode. Ce qui est donné par le tableau ci-dessous. Il est également rappelé la taille des procédés, présentée dans le tableau 4.5

	Nb. d'états	Nb. de transitions
Mode nominal N_1		
G^{N_1}	3 840	27 552
E^{N_1}	2 208	16 096
Mode nominal N_2		
G^{N_2}	23 040	219 072
E^{N_2}	14 592	111 744
Mode transitoire Mt_1		
G^{Mt_1}	13 440	132 352
E^{Mt_1}	18 432	193 664
Mode transitoire Mt_1		
G^{D}	6 720	59 456
E^{D}	4 608	28 448
Mode transitoire Mt_2		
G^{Mt_2}	6 720	59 456
E^{Mt_2}	1 024	7 936

TABLE 4.6 – Construction des modèles de spécifications E^{M_j}

4.5.3 Construction des procédés sous contrôle étendus

À partir des modèles des procédés G^{M_J} et des modèles de spécifications E^{M_J}, nous sommes en mesure de construire les procédés sous contrôle H^{M_j} et d'en évaluer leur taille. Le tableau ci-dessous donne ces valeurs.

H^{M_j}	Nb. d'états	Nb. de transitions
H^{N_1}	20 520	72 558
H^{N_2}	43 200	153 774
H^{D}	9 792	39 210
H^{Mt_1}	63 900	262 680
H^{Mt_2}	10 668	39 810

TABLE 4.7 – Construction des procédés sous contrôle H^{M_j}

4.5.4 Validation de l'étude intermodale et du suivi de trajectoire

Après plusieurs étapes successives d'étude intermodale, vérification du comportement des procédés sous contrôle, et suivi de trajectoire entre modèle afin de valider la consistance des modèles, nous avons convergé vers un point où il n'est plus nécessaire de toucher aux modèles car leur comportement a été validé, et sont consistants entre eux.

Nous sommes donc en mesure d'assurer que les commutation d'un mode à l'autre sont fiables et qu'il n'y a pas de perte d'informations lors de ces commutations. De même, si les commutations respectent les spécifications, c'est le cas également pour le comportement interne à chaque mode qui permet d'atteindre les objectifs voulus dans ceux-ci.

Nous donnons à titre d'informations dans le tableau ci-dessous le nombre de trajectoires de commutation trouvées et validées lors du suivi de trajectoire. Ce tableau mentionne pour chaque événement de commutation, le nombre de fois qu'il se produit pour activer ou désactiver un mode.

	$\tau^{M_j \to M_k}$: Nb. de transitions
avp	$\tau^{N_1 \to N_2}$: 4 644
ssp	$\tau^{N_2 \to N_1}$: 4 644
f_1	$\tau^{N_1 \to Mt_1}$: 1 161 $\tau^{N_2 \to Mt_1}$: 4 023
erb_3	$\tau^{Mt_1 \to D}$: 288
r_1	$\tau^{D \to Mt_2}$: 432
erb_4	$\tau^{Mt_2 \to N_1}$: 96

TABLE 4.8 – Nombre de transitions par événement de commutation

Le nombre de transitions où une panne peut survenir et faire commuter vers le mode Mt_1 est important. Cela est cohérent au regard de la taille du modèle représentant le comportement du mode Mt_1. Car pour s'assurer qu'aucune commutation non désirée ne puisse se produire, il n'y a que deux moyens : soit restreindre de manière drastique le procédé sous contrôle qui génère cette panne, soit générer beaucoup de comportement possible dans le/les modes cibles afin de prévoir toutes éventualités. Ceci doit être un choix défini et réalisé au plus tôt dans la conception.

4.6 Fusion des états non-significatifs

Les procédés sous contrôle ont une partie importante de leur comportement qui n'a été généré que pour valider les commutations entre les modèles. Cependant, les comportements qui nous intéressent sont ceux qui n'incluent que le comportement interne, et le comportement commutatif. Ce qui représentera le comportement modal d'un système, dans un mode. C'est la raison pour laquelle la dernière étape de la démarche est une étape de fusion des états non

significatifs, qui deviennent un état où le mode est inactif. Toutes les transitions sortantes de cet état activeront le mode, et toutes les transitions entrant dans cet état le désactiveront.

Le tableau suivant donne la taille pour chacun de modes du modèle représentant le comportement du système dans celui-ci.

$H_{merge}^{M_j}$	Nb. d'états	Nb. de transitions
$H_{merge}^{N_1}$	6 049	28 448
$H_{merge}^{N_2}$	26 137	96 057
$H_{merge}^{Mt_1}$	2 593	10 056
H_{merge}^{D}	1 945	6 264
$H_{merge}^{Mt_2}$	1 393	2 856

TABLE 4.9 – Construction des procédés sous contrôle fusionnés $H_{merge}^{M_j}$

4.7 Conclusion

Nous avons présenté dans cette section un exemple industriel de taille importe afin de montrer l'applicabilité de la démarche proposée. Ce système comprenait cinq modes, dont deux modes nominaux, correspondant à des objectifs différents, un mode dégradé, lié à une panne sur une machine, et à deux modes transitoires utilisés pour séparer des objectifs indépendants et faciliter la création des modèles de spécification.

À travers l'utilisation de la démarche proposée, nous avons dans un premier temps formalisé les exigences client en concevant les modèles mathématiques et vérifier sur ceux-ci certaines propriétés. Nous avons pu également, dans la première étude liée aux comportements internes dans les modes, vérifier le respect des spécifications représentant les objectifs de production dans chaque mode du système. Ensuite nous avons étendu ces comportements validés avec les commutations. De là, nous avons réalisé une succession de vérifications de contrôlabilité et de consistance sur ces commutations afin de valider le bon comportement des modèles de mode. Pour finir, nous avons réduit la taille des modèles pour ne garder que le comportement significatif à chaque mode. Ces derniers modèles, un par mode, représentent les comportements attendus et validés du système, et incluent toutes les commutations entrantes et sortantes admissibles.

Cet exemple a mis en avant la difficulté de prévoir toutes les commutations possibles autrement que par la génération de tous les comportements. Il est également fait remarquer qu'en utilisant la SCT, il est possible de restreindre le comportement générant ces commutations et donc de limiter le nombre de commutations à étudier entre modes. Ceci est une grande force de la démarche proposée. Les modèles fournis sont sûrs par construction, c'est-à-dire qu'ils respectent toutes les spécifications décrites, que ce soient des spécifications de sécurité, fonctionnelles ou liées aux commutations. Une attention toute particulière doit donc être portée par l'expert sur la construction des modèles des spécifications. Notamment pour s'assurer qu'elles

représentent bien les contraintes exprimées dans le cahier des charges.

Enfin, la taille réduite des modèles utilisés tout au long de cet exemple nous permet d'imaginer la possibilité d'utiliser cette démarche à des systèmes plus importants, notamment en restreignant plus fortement le comportement admissible, comme sur les spécifications de commutation, afin de ne pas générer de comportement inutile.

À travers cet exemple, les limites perçues se situent au niveau de l'impossibilité de calculer le langage menant à des états où un événement de commutation est généré. Ce langage est nécessaire pour effectuer un suivi de trajectoire et identifier une incompatibilité ou une inconsistance. Ceci est la seule étape qui est encore faite manuellement.

Conclusion générale

Le travail présenté dans ce mémoire concerne une démarche de conception appliquée à une gestion modale. Un mode est une configuration particulière du système où celui-ci exploite un ensemble de composants et doit respecter un ensemble de spécifications. L'utilisation de composants et le respect de spécifications donnent un comportement qui représente un fonctionnement temporaire du système dans le mode considéré.

La problématique de la gestion de mode concerne principalement l'élaboration des modes, leur activation/désactivation, et la possibilité de commuter entre eux.

Dans la première partie de ce mémoire, un état de l'art a mis en évidence certains points :

- La Théorie de Contrôle par Supervision est une approche formelle puissante pour la construction de modèles sûrs et évite des étapes de vérification des spécifications *a posteriori* sur les modèles créés.
- Il est primordial de réaliser une recherche exhaustive des commutations de mode au lieu de seulement définir celles autorisées. La création de tout ou partie de ces commutations, et la vérification de leur consistance, permettent d'éviter la perte d'information possible lors de commutations.
- Un nombre de modes restreint, et une prise en compte exhaustive des commutations, demandent une description du comportement de la partie commande concernant le comportement modal, et une description du comportement de la partie opérative pour la génération de tous les comportements commutatifs possibles. Les travaux ayant proposé une solution répondant à ces deux problèmes, nombre de modes et représentations exhaustives des commutations, sont ceux qui ont inclus ces deux descriptions.
- une approche multi-modèle, avec un modèle par mode, est adéquate pour représenter les comportements temporaires du système. Cependant, ces modèles fonctionnent de manière concurrente. Il faut donc s'assurer qu'ils ne peuvent pas générer de comportements contradictoires. Pour cela, l'utilisation d'un état inactif est pertinent. Cet état représente l'inactivité du mode, ou plutôt l'activation et la désactivation de celui-ci. Son utilisation assure que l'unicité de mode actif est respectée.

En plus de ces différents points, cette thèse fait suite aux travaux précédents concernant la

gestion des modes [Nou04, Kam05b]. Ces travaux ont mis en lumière certains résultats réutilisés par la suite comme : ajout d'un état inactif, possibilité de représenter un système à n modes ou suivi des traces pour éviter la perte d'information. Ils ont également mis en avant la suffisance d'un contrôleur par mode pour respecter à la fois les comportements interne et commutatif.

Cependant, ces travaux n'ont pas clairement défini les méthodes de construction des modèles, en accord avec la SCT, ni n'ont utilisé de spécifications sur les commutations. Les commutations étaient encore choisies et incluses manuellement par le concepteur, ce qui n'assurait pas que les spécifications de commutations soient respectées. De plus, réaliser cette opération d'extension sur des modèles de taille importante rend celle-ci improbable.

Ainsi, nous voulons que notre proposition, en utilisant la théorie de contrôle par supervision, soit une démarche complètement définie pour la construction des modèles de mode. Elle doit également garder des travaux précédents la séparation des différentes études suivant les comportements générés. Plus particulièrement, notre proposition doit proposer une démarche qui permette au concepteur de spécifier des contraintes sur les commutations du système tout en assurant une contrôlabilité sur celui-ci. La démarche doit permettre d'obtenir un modèle par mode où peut être extraite la loi de commande à implémenter dans le système physique.

Nous avons répondu à ces objectifs en proposant une démarche de conception constituée de 5 étapes. Chacune des étapes répond à des problèmes sur la gestion des modes, et à un ou plusieurs des objectifs exposés. La première étape est la formalisation des exigences client où nous retranscrivons par des modèles formels, automates à états, le cahier des charges. La deuxième étape, celle de l'étude intramodale, concerne la construction et la validation du comportement interne à chaque mode à l'égard des exigences du client et indépendamment du comportent des autres modes. Il convient d'accorder dans cette étape une attention particulière à la modélisation des spécifications. La troisième étape, l'étude intermodale, étend les modèles validés à l'étape 2 avec le comportement commutatif du mode et vérifie que les spécifications sur les commutations sont respectées. Seules les commutations autorisées sont représentées dans les modèles issus de cette étape. Cependant, rien n'assure qu'une commutation dans un mode conduise de manière effective à un autre mode. Ce dernier peut très bien ne pas avoir le comportement menant à cette commutation. L'étape 4 vérifie ainsi, par une procédure de suivi de trajectoire, que toutes les commutations qui désactivent un mode activent également un autre mode. Le système est toujours dans un état et un mode déterminé. Nous qualifions cela de consistance entre les modèles de modes. Enfin, la possibilité d'avoir un seul modèle par mode implique que les modèles représente des comportements qui fonctionnent de manière concurrente et qui, pris ensemble, représentent un comportement équivalent à un seul modèle monolithique. La dernière étape fusionne les états non significatifs des modèles et crée un état inactif. Cet état représente l'inactivité du mode et assure un fonctionnement concurrent entre modèles.

Au-delà de la démarche, nous avons également montré l'équivalence de comportement entre modèles concurrents construits séparément et un modèle monolithique construit par approche

classique. Cette équivalence de comportement est effective si les modèles ont été construits à partir de spécifications identiques.

Nous avons fourni une démarche claire et définie pour représenter formellement, concevoir méthodiquement et vérifier mathématiquement les modes et leurs commutations. Les modèles sont sûrs par construction et toutes les commutations représentées ont été vérifiées. Dans cette démarche, toutes les commutations sont étudiées et seules celles assurant un changement de mode sûr ont été autorisées. L'approche multi-modèle permet également de maîtriser l'explosion combinatoire par des modèles distincts tout en permettant une équivalence de comportement avec une approche monolithique.

Une perspective à nos travaux serait l'automatisation complète de cette démarche pour la rendre plus facile d'accès aux concepteurs. Des étapes ont déjà été réalisées, cependant quelques étapes nécessitent encore un expert pour être exploitées. Une autre perspective à ses travaux concerne une extension de l'automate de modes et plus particulièrement la possibilité de le rendre hiérarchique afin d'avoir un système à plusieurs niveaux de fonctionnement où chaque niveau aurait différents modes de fonctionnement.

A

Table de notations

Nous proposons dans cette annexe un tableau rappelant les noms des modèles ou les noms des fonctions, et ce qu'ils représentent. L'intérêt de ce tableau est d'être utilisé pendant la lecture de cette thèse pour avoir en permanence sous la main la signification des notations utilisées.

Notation	Signification
$G = (Q, \Sigma, \delta, q_0, Q_m)$	Notation mathématique pour représenter un automate
Q	Ensemble d'états
$\Sigma = \Sigma_{\circlearrowleft} \cup \Sigma_{\leftrightarrows}$	Ensemble des événements
$\Sigma_{\circlearrowleft} \subseteq \Sigma$	Ensemble des événements internes
$\Sigma_{\leftrightarrows} \subset \Sigma$	Ensemble des événements de commutation
δ	Fonction de transition
$\delta^{\leftrightarrows} \in \delta$	Ensemble des fonctions de transition définissant une commutation de mode
q_0	État initial de l'automate
Q_m	Ensemble des états finaux ou marqués
$E = (X, \Sigma, \xi, x_0, X_m)$	Modèle des spécifications.
$K = G \times E$	Modèle du procédé sous contrôle désiré.
$H = K$	Modèle du procédé sous contrôle si K est contrôlable par rapport à G suivant le théorème 2.
$H = K^{\uparrow c}$	Modèle du procédé sous contrôle si K n'est pas contrôlable. Il est calculé par le suprême contrôlable défini section 1.3.3.
$\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$	ensemble de composants.
$\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$	ensemble de modes.
$\mathcal{C}^{M_j} = \mathcal{C}_{\circlearrowleft}^{M_j} \cup \mathcal{C}_{\leftarrow}^{M_j} \cup \mathcal{C}_{\rightarrow}^{M_j}$	ensemble de composants dans M_j, avec $(\mathcal{C}_{\circlearrowleft}^{M_j})$ l'ensemble des composants utilisés, $(\mathcal{C}_{\leftarrow}^{M_j})$ l'ensemble des composants générant un événement activant le mode, et $\mathcal{C}_{\rightarrow}^{M_j}$ l'ensemble des composants générant un événement désactivant le mode.
$G^{C_i} = (Q^{C_i}, \Sigma^{C_i}, \delta^{C_i}, q_0^{C_i}, Q_m^{C_i})$	Modèle d'un composant.
$G^{\mathcal{M}} = (Q^{\mathcal{M}}, \Sigma^{\mathcal{M}}, \delta^{\mathcal{M}}, q_0^{\mathcal{M}}, Q_m^{\mathcal{M}})$	Modèle de l'automate de modes. Il représente le comportement commutatif du système.
Modèles de l'étude intramodale	
$G_{in}^{M_j} = (Q_{in}^{M_j}, \Sigma_{in}^{M_j}, \delta_{in}^{M_j}, q_{in,0}^{M_j}, Q_{in,m}^{M_j})$	Modèle du procédé représentant le comportement interne dans le mode M_j.
$E_{in}^{M_j}$	Modèle représentant les spécifications que le comportement interne doit respecter dans le mode M_j.
$H_{in}^{M_j} = (Y_{in}^{M_j}, \Sigma_{in}^{M_j}, \tau_{in}^{M_j}, y_{in,0}^{M_j}, Y_{in,m}^{M_j})$	Modèle du procédé sous contrôle représentant le comportement interne qui respecte les spécifications dans le mode M_j.
Modèle de l'étude intermodale	
$G^{M_j} = (Q^{M_j}, \Sigma^{M_j}, \delta^{M_j}, q_0^{M_j}, Q_m^{M_j})$	Modèle du procédé étendu représentant le comportement complet du système dans le mode M_j.
$E^{M_j} = E_{\circlearrowleft}^{M_j} \times E_{\rightleftarrows}^{M_j}$	Modèle représentant les spécifications intermodales à respecter dans le mode M_j
$E_{\circlearrowleft}^{M_j}$	Modèle des spécifications intramodales étendues.
$E_{\rightleftarrows}^{M_j}$	Modèle des spécifications de commutations.
$H^{M_j} = (Y^{M_j}, \Sigma^{M_j}, \tau^{M_j}, y_0^{M_j}, Y_m^{M_j})$	Modèle du procédé sous contrôle représentant le comportement qui respecte les spécifications dans le mode M_j.
$Y_{M_j \xrightarrow{\alpha} M_k}^{M_j}$	Ensemble des états appartenant à H^{M_j} où un événement de commutation α existe et implique une commutation du mode de départ M_j vers le mode d'arrivée M_k.
$L_{M_j \xrightarrow{\alpha} M_k}^{y}(H^{M_j})$	Sous-langage de $L(H^{M_j})$, qui contient tous les mots menant de l'état initial y_0 de H^{M_j} vers un état y où un événement de commutation α peut se produire.
$P_{M_j,M_k} : \Sigma^{M_j *} \rightarrow \Sigma^{M_k *}$	Fonction de projection étendue. Elle se base sur la fonction de projection définie section 1.2.2, étendue sur deux ensembles d'événements où l'un n'est pas contenu dans l'autre.
$H_{lab}^{M_j} = (Y^{M_j}, \Sigma_{lab}^{M_j}, \tau_{lab}^{M_j}, y_0^{M_j}, Y_m^{M_j})$	Modèle du procédé sous contrôle après la procédure de suivi de trajectoire 2 définie section 3.7.2.
$Y_{mer}^{M_j} \subset Q_{lab}^{M_j}$	Ensemble des états non significatifs que nous souhaitons fusionner.
$y_{id}^{M_j}$	État inactif résultant de la fusion des états non significatifs.
$H_{merge}^{M_j} = (Y_{merge}^{M_j}, \Sigma_{merge}^{M_j}, \tau_{merge}^{M_j}, y_{merge,0}^{M_j}, Y_{merge,m}^{M_j})$	Modèle du procédé sous contrôle fusionné dans le mode M_j. Le comportement restant est le comportement interne et le comportement commutatif.

TABLE A.1 – Table de notations

Bibliographie

[ADE81] ADEPA. Guide d'etude des modes de marches et d'arrêts. Technical report, ADEPA, France, 1981.

[Ake03] K. Akesson, M. Fabian, H. Flordal, et A Vahidi. Supremica - a tool for verification and synthesis of discrete event supervisors. In *Proc. of the 11th Mediterranean Conference on Control and Automation*, vol. 238, pages 3–19, 2003. Elsevier Science Publishers B. V.

[Ake06] K. Akesson, M. Fabian, H. Flordal, et R. Malik. Supremica - an integrated environment for verification, synthesis and simulation of discrete event systems. In *Discrete Event Systems, 8th International Workshop on*, pages 384 –385, juillet 2006.

[And96] C. André. Syncharts : A visual representation of reactive behaviors. Technical report, I3S Laboratory - ESTEREL Technologies, avril 1996.

[Bal00] A. Balluchi, L. Benvenuti, M.D. di Benedetto, C. Pinello, et A.L. Sangiovanni-Vincentelli. Automotive engine control and hybrid systems : challenges and opportunities. *Proceedings of the IEEE*, 88(7) :888 –912, jul 2000.

[Ber92] G. Berry et G. Gonthier. The esterel synchronous programming language : Design, semantics, implementation. *Science of Computer Programming*, 19(2) :87–152, November 1992.

[Car07] O. Carton. *Langages formels, Calculabilité et Complexité*. Éditions Vuibert, 2007.

[Cas87] P. Caspi, D. Pilaud, N. Halbwachs, et J. A. Plaice. Lustre : a declarative language for real-time programming. In *Principles of programming languages, POPL'87*, pages 178–188, NY, USA, 1987.

[Cas07] C.G. Cassandras et S. Lafortune. *Introduction to discrete event systems [Second Edition]*. Springer, 2007.

[Cha00] S. Chafik et E. Niel. Hierarchical-decentralized solution of supervisory control. In *3rd International Symposium on Mathematical Modelling, 3rd MATHMOD*, vol. 2, pages 787–790, Vienne, Autriche, 2000.

[Cha03] Z. Chao et Y. Xi. Necessary conditions for control consistency in hierarchical control of discrete-event systems. *IEEE Transactions on Automatic Control*, 48 :465–468, mars 2003.

[Cho59] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2) :137 – 167, 1959.

[Cra94] E. Craye. Contribution au contrôle/commande de systèmes flexibles de production manufacturière. Rapport Technique, Université des Sciences et Technologies de Lille, 1994.

[Dan00] N. Dangoumau. *Contributions à la Gestion des Modes des Systèmes Automatisés de Production*. Thèse de doctorat, Ecole Centrale de Lille, décembre 2000.

[Dan02] N. Dangoumau, A.K.A. Toguyéni, et E. Craye. Functional and behavioural modelling for dependability in automated production systems. *Proceedings of the Institution of Mechanical Engineers, Part B : Journal of Engineering Manufacture*, 216(3) :389–405, 2002.

[Far08] G. Faraut, L. Piétrac, et E. Niel. Identification of incompatible states in mode switching. In *Emerging Technologies and Factory Automation. ETFA'08. IEEE International Conference on*, pages 121–128, sept. 2008.

[Far09a] G. Faraut, L. Piétrac, et E. Niel. Démarche d'aide à la conception par approche multimode des sed. *Journal Européen des Systèmes Automatisés*, 43(7-9) :837–853, 2009.

[Far09b] G. Faraut, L. Pietrac, et E. Niel. Formal approach to multimodal control design : Application to mode switching. *Industrial Informatics, IEEE Transactions on*, 5(4) :443–453, nov. 2009.

[Far09c] G. Faraut, L. Piétrac, et E. Niel. A new framework for mode switching in sct. In *European Control Conference, ECC'09*, août 2009.

[Fen08] L. Feng et W.M. Wonham. Supervisory control architecture for discrete-event systems. *IEEE Transactions on Automatic Control*, 53(6) :1449–1461, 2008.

[Har87] D. Harel. Statecharts : A visual formalism for complex systems. *Science of Computer Programming*, 8(3) :231–274, juin 1987.

[Hal91] N. Halbwachs, P. Caspi, P. Raymond, et D. Pilaud. The synchronous data flow programming language lustre. *Proceedings of the IEEE*, 79(9) :1305 –1320, sept. 1991.

[Ham04] N. Hamani, N. Dangoumau, et E. Craye. A formal approach for reactive mode handling. In *Systems, Man and Cybernetics, IEEE International Conference on*, vol. 5, pages 4306–4311, oct. 2004.

[Ham06] N. Hamani, N. Dangoumau, et E. Craye. Spécification et vérification pour la gestion des modes des systèmes automatisés de production : analyse des besoins et approches existantes. In *Modélisation, Optimisation et Simulation des Systèmes : Défis et Opportunités, MODSIM*, 2006.

[Ham09] N. Hamani, N. Dangoumau, et E. Craye. Verification and validation of a ssm model dedicated to mode handling of flexible manufacturing systems. *Computers in Industry*, 60(2) :77–85, 2009.

[Hil08] R. Hill. *Modular Verification and Supervisory Controller Design for Discrete-Event Systems Using Abstraction and Incremental Construction*. Thèse de doctorat, Université du Michigan, 2008.

[Hil10] R. Hill, D. Tilbury, et S. Lafortune. Modular supervisory control with equivalence-based abstraction and covering-based conflict resolution. *Discrete Event Dynamic Systems*, 20(1) :139–185, 2010.

[Jah88] F. Jahanian et D.A. Stuart. A method for verifying properties of modechart specifications. In *In Proceedings of the Real-Time Systems Symposium*, pages 12–21, 1988.

[Jah94] F. Jahanian et A.K. Mok. Modechart : A specification language for real-time systems. *IEEE Trans. Softw. Eng.*, 20(12) :933–947, 1994.

[Jia00] S. Jiang et R. Kumar. Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30(5) :653–660, October 2000.

[Kam04] O. Kamach. *Approche multi-modèle pour les systèmes à événements discrets : application à la gestion des modes de fonctionnement*. Thèse de doctorat, INSA de Lyon, déc. 2004.

[Kam05a] O. Kamach, L. Piétrac, and E. Niel. Supervisory uniqueness for operating mode systems. In *16th IFAC world congress*, Prague, 4–8 July 2005.

[Kam05b] O. Kamach, L. Piétrac, et Eric Niel. Multi-model approach to discrete events systems : Application to operating mode management. *Mathematics and Computers in Simulation, Elsevier*, 70(5-6) :394–407, 2005.

[Kam06a] O. Kamach, L. Piétrac, et E. Niel. Design of switching supervisors for reactive class discrete event systems. *Information Control Problems in Manufacturing 2006*, 12(1) :289–294, 2006.

[Kam06b] O. Kamach, L. Piétrac, et E. Niel. Forbidden and preforbidden states in the multi-model approach. *Computational Engineering in Systems Applications, IMACS Multiconference on*, 2 :1550–1557, oct. 2006.

[Kha99] A. Khatab et E. Niel. Predicates and predicate transformers for the supervisory control of timed discrete event system. In *Emerging Technologies and Factory Automation. ETFA'99. 7th IEEE International Conference on*, vol. 2, pages 1081–1090, 1999.

[Kha02] A. Khatab et E. Niel. State feedback stabilizing controller for the failure recovery of timed discrete event systems. In *Proceedings of the Sixth International Workshop on Discrete Event Systems, WODES'02*, pages 113–118, 2002.

[Kom08] J. Komenda, J.H. Van Schuppen, B. Gaudin, et H. Marchand. Supervisory control of modular systems with global specification languages. *Automatica*, 44(4) :1127–1134, avril 2008.

[Koo01] T.J. Koo, G.J. Pappas, et S. Sastry. Mode switching synthesis for reachability specifications. In *Hybrid Systems : Computation and Control*, vol. 2034, pages 333–346. Springer Berlin / Heidelberg, 2001.

[Koo01a] T.J. Koo, G.J. Pappas, et S. Sastry. Multi-modal control of systems with constraints. In *Decision and Control. Proceedings of the 40th IEEE Conference on*, vol. 3, pages 2075 –2080, 2001.

[LeG91] P. Le Guernic, T. Gautier, M. Le Borgne, et C. Le Maire. Programming real-time applications with signal. *Proceedings of the IEEE*, 79(9) :1321 –1336, sep 1991.

[Lin05] G. Lindstrom. Programming with python. *IT Professional*, 7(5) :10 – 16, sep. 2005.

[Mar92] F. Maraninchi. Operational and compositional semantics of synchronous automaton compositions. In *International Conference on Concurrency Theory*, pages 550–564, 1992.

[Mar98] F. Maraninchi et Y. Rémond. Mode-automata : About modes and states for reactive systems. *Lecture Notes in Computer Science*, 1381/1998 :105–115, 1998.

[Mar03] F. Maraninchi et Y. Rémond. Mode-automata : a new domain-specific construct for the development of safe critical systems. *Science of Computer Programming*, 1(46) :219–254, mars 2003.

[McM92] K.L. McMillan. *Symbolic Model Checking : an approach to the state explosion problem*. Thèse de doctorat, Université de Carnegie Mellon, 1992.

[Mor97] S. Moreno et E. Peulot. *Le GEMMA [Texte imprimé] : modes de marches et d'arrêts, GRAFCET de coordination des tâches, conception des systèmes automatisés de production sûrs : bac STI, BTS, DUT, IUP, écoles d'ingénieurs*. Éducalivre-Casteilla, 1997, 250p.

[Mos00] P.J. Mosterman et H. Vangheluwe. Computer automated multi-paradigm modeling in control system design. *IEEE Transactions on Control System Technology*, 12 :65–70, 2000.

[Mur89] T. Murata. Petri nets : Properties, analysis and applications. *Proceedings of the IEEE*, 77(4) :541 –580, avril 1989.

[Nie93] E. Niel, P. Burgat, B. Mille, et A. Jutard. A contribution to the operational safety evaluation by use of a misfunctional modeling approach. In *Systems, Man and Cybernetics. 'Systems Engineering in the Service of Humans', International Conference on*, vol.1, pages 389–394, oct. 1993.

[Nie95] E. Niel, B. Brandin, S. Boukhobza, et M. Nourelfath. Operational-safety supervisory control : an approach to supervisor activation. In *Emerging Technologies and Factory Automation, ETFA'95*, vol. 2, pages 553–561, oct. 1995.

[Nie96] E. Niel, N. Rezg, M. Nourelfath, et S. Boukhobza. Supervisory control in the context of operational safety. In *Computational Engineering in Systems Applications, CESA'96*, pages 746–751, 9-12 juillet 1996.

[Nou96] M. Nourelfath, E. Niel, et S. Boukhobza. Formalisation d'un module de surveillance à base de la théorie des automates : synthèse et implantation sur un atelier flexible. *Modélisation des Systèmes Réactifs, MSR'96*, pages 267–275, mars 1996.

[Nou96a] M. Nourelfath et E. Niel. Formal specification and synthesis of discrete event systems control and supervision. In *Computational Engineering in Systems Applications, CESA'96*, pages 267–272, juillet 1996.

[Nou96b] M. Nourelfath et E. Niel. On the use of macroactions in supervisory control theory for monitoring purpose. In *Emerging Technologies and Factory Automation, EFTA'96*, vol. 1, pages 157–162, 1996.

[Nou97] M. Nourelfath. *Extension de la théorie de la supervision à la surveillance et à la commande des systèmes à événements discrets : application à la sécurité opérationnelle des systèmes de prodcution.* Thèse de doctorat, INSA de Lyon, 1997.

[Nou04] M. Nourelfath et E. Niel. Modular supervisory control of an experimental manufacturing system. *Control Engineering Pratice*, 12(2) :205–216, 2004.

[Now96] G. Nowak, E. Niel, et A. Jutard. Specification method based on temporal sadt and deterministic/stochastic petri nets. application to safety assessment. In *Emerging Technologies and Factory Automation, EFTA'96*, vol. 1, pages 163–166, nov. 1996.

[Oli07] T.E. Oliphant. Python for scientific computing. *Computing in Science Engineering*, 9(3) :10 –20, 2007.

[Pet81] J.L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[Puc95] C. Puchol, A. K. Mok, et D.A. Stuart. Compiling modechart specifications. In *IEEE Real-Time Systems Symposium*, pages 256–265, 1995.

[Que05] M.H. Queiroz, J.E. Cury, et W.M. Wonham. Multitasking supervisory control of discrete-event systems. *Discrete Event Dynamic Systems*, 15(4) :375–395, 2005.

[Rad97] J. Radatz. *The IEEE Standard Dictionary of Electrical and Electronics Terms*. IEEE Standards Office, New York, NY, USA, 1997.

[Rak05] E. Rakotomalala. *Spécifications Robustes du Système de Pilotage d'une Fonction Electronique Automobile*. Thèse de doctorat, Ecole Centrale de Nantes, 2005.

[Ram87] P.J. Ramadge et W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1) :206–230, 1987.

[Ram89] P.J. Ramadge et W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1) :81–98, 1989.

[Rez93] N. Rezg, E. Eric Niel, et A. Jutard. Modélisation structurelle des systèmes automatisés de production. In *13èmes Journées Tunisiennes d'Electrotechnique et d'Automatique, JTEA'93*, 1993.

[Rez95] N. Rezg et E. Niel. Monitoring system for discrete event systems using failure-tolerance techniques. In *Emerging Technologies and Factory Automation, ETFA'95*, vol. 1, pages 383–391, oct. 1995.

[Ric06] L. Ricker, S. Lafortune, et S. Gene. Desuma : A tool integrating giddes and umdes. In *Discrete Event Systems, 8th International Workshop on*, pages 392 –393, juillet 2006.

[Rud92] K. Rudie et W.M. Wonham. Think globally, act locally : decentralized supervisory control. *IEEE Transactions on automatic control*, 37 :1692–1708, 1992.

[Sch07] K. Schmidt, M.H. Queiroz, et J.E. Cury. Hierarchical and decentralized multitasking control of discrete event systems. In *Decision and Control, 46th IEEE Conference on*, pages 5936–5941, déc. 2007.

[Sip05] M. Sipser. *Introduction to the Theory of Computation, Second Edition*. Course Technology, fév. 2005.

[Sto05] G. Stoneburner. Developer-focused assurance requirements [evaluation assurance level and common criteria for it system evaluation]. *Computer*, 38(7) :91–93, 2005.

[Su04] R. Su et W.M. Wonham. Supervisor reduction for discrete-event systems. *Discrete Event Dynamic Systems : theory and applications*, 14 :31–53, 2004.

[Tak98] S. Takai. On the languages generated under fully decentralized supervision. *IEEE Transactions on Automatic Control*, 43(9) :1253–1256, 1998.

[Vie06] A.D. Vieira, J.E. Cury, et M.H. Queiroz. A model for plc implementation of supervisory control of discrete event systems. In *Emerging Technologies and Factory Automation, ETFA '06*, pages 225–232, 2006.

[Won87] W.M. Wonham et P.J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, 25(3) :637–659, 1987.

[Won09] W.M. Wonham. Supervisor control of discrete-event systems ece 1636f/1637s 2009-10. course notes, departement of Electrical and Computer Engineering, Univeristé de Toronto, 2009.

[Yoo02] S. Yoo et S. Lafortune. Decentralized supervisory control : a new architecture with a dynamic decision fusion rule. In *6th international Workshop On Discrete Event Systems, WODES'02*, pages 11–17, Saragosse, Espagne, oct. 2002.

[Zay99] J. Zaytoon, C Ndjab Hagbebell, et V. Carre-Menetrier. Grafcet et graphes d'états : synthèse hors ligne de la commande. In *Journal Européen des Systèmes Automatisés*, 1999.

[Zho90] H. Zhong et W.M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *Automatic Control, IEEE Transactions on*, 35(10) :1125–1134, oct. 1990.

Liste des figures

Liste des définitions, théorèmes et lemmes

Définitions

Théorèmes

Liste des tableaux

Commutations sûrs de mode pour les systèmes à événements discrets

Résumé : Le travail présenté dans ce mémoire concerne une démarche de conception appliquée à une gestion modale pour les systèmes à événements discrets (SED). Un mode est une configuration particulière du système où celui-ci exploite un ensemble de composants et doit respecter un ensemble de spécifications. La problématique de la gestion de mode porte principalement sur la conception des modes et sur leurs commutations. Notre objectif est de proposer une démarche de conception complètement définie où les spécifications sont assurément respectées, et où seules les commutations désirées entre modes peuvent se produire. Il est également vérifié que toute commutation dans un mode mène de manière sûre dans un autre mode. Pour réaliser cet objectif, nous utilisons la théorie de contrôle par supervision qui permet de concevoir des modèles sûrs par construction tel que les spécifications utilisées pour la construction soient respectées. La démarche proposée possède plusieurs étapes séparant ainsi les différentes études de conception. La première concerne la formalisation du cahier des charges en modèles automate à états. L'étude suivante concerne le comportement interne où celui-ci doit respecter les spécifications propres aux modes, indépendamment des autres modes. Cette étape valide le comportement de chaque mode, avant d'étudier leurs commutations. La troisième étape étudie le comportement commutatif tel que les spécifications de commutations soient respectées. Cette étape spécifie les commutations désirées, et inversement celles non voulues. L'étape suivante est l'exécution d'une fonction de suivi de trajectoire qui vérifie que toutes les commutations mènent bien dans un autre mode. Dans le cas contraire, la fonction de suivi identifie et caractérise les commutations problématiques afin d'aider le concepteur dans la résolution de ces situations. Enfin, une étape de fusion d'états finalise la démarche afin de fournir un modèle par mode qui représente le comportement de celui-ci. Pour montrer l'applicabilité de la démarche proposée, et sa faculté à être utilisée en milieu industriel, nous l'utilisons sur un exemple de taille importante utilisée dans la littérature.

Mots clés : Approche modale, Théorie de contrôle par supervision, Automate à états, Gestion de modes, Système à événements discrets, Théorie des langages, Synthèse de contrôleurs, Reconfiguration.

Safe switching of mode for discrete-event systems

Abstract : The work presented in this thesis concerns a framework applied to the modal approach for discrete-event system (DES). A mode is a particular configuration of a system where this one handles a set of components and has to respect a set of specifications. The problem of mode management is about the design of modes and on their switching. The aim of our work is to propose a, completely defined, designing framework, where the specifications are guaranteed, and only the admissible switching between modes may happen. It also verifies that every switching in a mode effectively leads to safely another one. To reach this goal, we firstly use the Supervisory Control Theory (SCT) which computes safe models in which the requirements are respected. The proposed framework is composed by several steps, splitting the different studies of design. The first step focuses on the formalization of requirements into mathematical models - Finite State Machine. The next one concerns the synthesis by SCT of the internal behavior of mode to ensure that the specifications are respected, independently of other modes. The third step studies the switching behavior such as the switching specifications in each mode are respected. In this step, the admissible switchings are specified and forbid the other ones. The next step is a function of process tracking which verifies that all switchings effectively lead into only one mode. In the opposite case, the function of process tracking identifies and characterizes the problematic switchings to help the designer to solve these possibilities. At the end, a step of merging states is computed to remove the non-significant states and obtaining one model per mode representing the behavior of this one. To show the applicability of the proposed framework, we apply it on an example used in the literature.

Key words : Modal approach, Supervisory control theory, Language theory, Automata, Modes management, Discrete-event systems, Reconfiguration, Controller synthesis.